





FRK: An R Package for Spatial and Spatio-Temporal Prediction with Large Datasets

Andrew Zammit-Mangion 
University of Wollongong

Noel Cressie 
University of Wollongong

Abstract

FRK is an R software package for spatial/spatio-temporal modeling and prediction with large datasets. It facilitates optimal spatial prediction (kriging) on the most commonly used manifolds (in Euclidean space and on the surface of the sphere), for both spatial and spatio-temporal fields. It differs from many of the packages for spatial modeling and prediction by avoiding stationary and isotropic covariance and variogram models, instead constructing a spatial random effects (SRE) model on a fine-resolution discretized spatial domain. The discrete element is known as a basic areal unit (BAU), whose introduction in the software leads to several practical advantages. The software can be used to (i) integrate multiple observations with different supports with relative ease; (ii) obtain exact predictions at millions of prediction locations (without conditional simulation); and (iii) distinguish between measurement error and fine-scale variation at the resolution of the BAU, thereby allowing for reliable uncertainty quantification. The temporal component is included by adding another dimension. A key component of the SRE model is the specification of spatial or spatio-temporal basis functions; in the package, they can be generated automatically or by the user. The package also offers automatic BAU construction, an expectation-maximization (EM) algorithm for parameter estimation, and functionality for prediction over any user-specified polygons or BAUs. Use of the package is illustrated on several spatial and spatio-temporal datasets, and its predictions and the model it implements are extensively compared to others commonly used for spatial prediction and modeling.

Keywords: basic areal units, EM algorithm, fixed rank kriging, spatial random effects model, spatial prediction.

1. Introduction

Fixed rank kriging (FRK) is a spatial/spatio-temporal modeling and prediction framework that is scaleable, works well with large datasets, and can deal easily with data that have different spatial supports. FRK hinges on the use of a spatial random effects (SRE) model, in

which a spatially correlated mean-zero random process is decomposed using a linear combination of spatial basis functions with random coefficients plus a term that captures the random process' fine-scale variation. Dimensionality reduction through a relatively small number of basis functions ensures computationally efficient prediction, while the reconstructed spatial process is, in general, non-stationary. The SRE model has a spatial covariance function that is always nonnegative-definite and, because any (possibly non-orthogonal) basis functions can be used, it can be constructed so as to approximate standard families of covariance functions (Kang and Cressie 2011). For a detailed treatment of FRK, see Cressie and Johannesson (2006, 2008), Shi and Cressie (2007), and Nguyen, Cressie, and Braverman (2012).

There are numerous R (R Core Team 2021) packages available for modeling and prediction with spatial or spatio-temporal data (see Bivand 2021), although relatively few of these make use of a model with spatial basis functions. A few variants of FRK have been developed to date, and the one that comes closest to the present software is **LatticeKrig** (Nychka, Bandyopadhyay, Hammerling, Lindgren, and Sain 2015; Nychka, Hammerling, Sain, Lenssen, Smirniotis, and Iverson 2019). **LatticeKrig** implements what we call a LatticeKrig (LTK) model, which is made up of Wendland basis functions (that have compact support) decomposing a spatially correlated process. LatticeKrig models use a Markov assumption to construct a precision matrix (the matrix \mathbf{K}^{-1} in Section 2.1) to describe the dependence between the coefficients of these basis functions. This, in turn, results in efficient computations and the potential use of a large number ($> 10,000$) of basis functions. LatticeKrig models do not cater for what we term fine-scale-process variation and, instead, the finest scale of the process is limited to the finest resolution of the basis functions used.

The package **INLA** (Lindgren and Rue 2015) is a general-purpose package for model fitting and prediction. One advantage of **INLA** is that it contains functionality for fitting Gaussian processes that have covariance functions from the Matérn class (see Lindgren and Rue 2015, for details on the software interface) by approximating a stochastic partial differential equation (SPDE) using a Gaussian Markov random field (GMRF). Specifically, the process is decomposed using basis functions that are triangular 'tent' functions, and the coefficients of these basis functions are normally distributed with a sparse precision matrix. Thus, these models, which we term SPDE-GMRF models, share many of the features of LatticeKrig models. A key advantage of **INLA** over **LatticeKrig** is that once the spatial or spatio-temporal model is constructed, one has access to all the approximate-inference machinery and likelihood models available within the package.

Kang and Cressie (2011) develop Bayesian FRK; they keep the spatial basis functions fixed and put a prior distribution on \mathbf{K} . The predictive-process approach of Banerjee, Gelfand, Finley, and Sang (2008) can also be seen as a type of Bayesian FRK, where the basis functions are constructed from the postulated covariance function of the spatial random effects and hence depend on parameters (see Katzfuss and Hammerling 2017, for an equivalence argument). An R package that implements predictive processes is **spBayes** (Finley, Banerjee, and Carlin 2007; Finley, Banerjee, and Gelfand 2015). It allows for multivariate spatial or spatio-temporal processes, and Bayesian inference is carried out using Markov chain Monte Carlo (MCMC), thus allowing for a variety of likelihood models. Because the implied basis functions are constructed based on a parametric covariance model, a prior distribution on parameters results in new basis functions generated at each MCMC iteration. Since this can slow down the computation, the number of knots used in predictive processes is usually chosen to be small, which has the effect of limiting their ability to model finer scales.

Our software package **FRK** differs from spatial prediction packages currently available by constructing an SRE model on a discretized domain, where the discrete element is known as a basic areal unit (BAU; see, e.g., [Nguyen *et al.* 2012](#)). The BAU can be viewed as the smallest spatial area or spatio-temporal volume that can be resolved by the process and, to reflect this, the process itself is assumed to be piecewise constant over the set of BAUs. The BAUs serve many purposes in **FRK**: They define a fine grid over which to do numerical integrations for change-of-support problems; a fine lattice of discrete points over which to predict (although **FRK** implements functions to predict over any arbitrary user-defined polygons); and a set of bins within which to average large spatio-temporal datasets, if so desired, for computational efficiency. BAUs do not need to be square or all equal in size, but they do need to be ‘small,’ in the sense that they should be able to reconstruct the (undiscretized) process with minimal error.

In the standard ‘flavor’ of FRK ([Cressie and Johannesson 2008](#)), which we term vanilla FRK (FRK-V), there is an explicit reliance on multi-resolution basis functions to give complex non-stationary spatial patterns at the cost of not imposing any structure on \mathbf{K} , the covariance matrix of the basis function weights. This can result in identifiability issues and hence in over-fitting the data when \mathbf{K} is estimated using standard likelihood methods (e.g., [Nguyen, Katzfuss, Cressie, and Braverman 2014](#)), especially in regions of data paucity. Therefore, in **FRK** we also implement a model (FRK-M) where a parametric structure is imposed on \mathbf{K} (e.g., [Stein 2008](#); [Nychka *et al.* 2015](#)). The main aim of the package **FRK** is to facilitate spatial and spatio-temporal analysis and prediction for large datasets, where multiple observations come with different spatial supports. We see that in ‘big data’ scenarios, lack of consideration of fine-scale variation may lead to over-confident predictions, irrespective of the number of basis functions adopted.

In Section 2, we describe the modeling, estimation, and prediction approach we adopt in **FRK**. In Section 3, we discuss further details of the package and provide a simple example on the classic `meuse` dataset. In Section 4, we evaluate the SRE model implemented in **FRK** in controlled cases, against LatticeKrig models and SPDE-GMRF models through use of the packages **LatticeKrig** and **INLA**. In Section 5, we show its capability to deal with change-of-support issues and anisotropic processes. In Section 6, we show how to use **FRK** with spatio-temporal data and illustrate its use on the modeling and prediction of column-averaged carbon dioxide on the globe from remote sensing data produced by NASA’s OCO-2 mission. The spatio-temporal dataset contains millions of observations. Finally, Section 7 discusses future work.

2. Outline of FRK: Modeling, estimation and prediction

In this section we present the theory behind the operations implemented in **FRK**. In Section 2.1 we introduce the SRE model, in Section 2.2 we discuss the EM algorithm for parameter estimation, and in Section 2.3 we present the spatial prediction equations.

2.1. The SRE model

Denote the spatial process of interest as $\{Y(\mathbf{s}) : \mathbf{s} \in D\}$, where \mathbf{s} indexes the location of $Y(\mathbf{s})$ in our domain of interest D . In what follows, we assume that D is a spatial domain but extensions to spatio-temporal domains are natural within the framework (Section 6).

Consider the classical spatial statistical model,

$$Y(\mathbf{s}) = \mathbf{t}(\mathbf{s})^\top \boldsymbol{\alpha} + v(\mathbf{s}) + \xi(\mathbf{s}); \quad \mathbf{s} \in D,$$

where, for $\mathbf{s} \in D$, $\mathbf{t}(\mathbf{s})$ is a vector of spatially referenced covariates, $\boldsymbol{\alpha}$ is a vector of regression coefficients, $v(\mathbf{s})$ is a small-scale, spatially correlated random effect, and $\xi(\mathbf{s})$ is a fine-scale random effect that is ‘almost’ spatially uncorrelated. It is natural to let $\mathbf{E}(v(\cdot)) = \mathbf{E}(\xi(\cdot)) = 0$. Define $\lambda(\cdot) \equiv v(\cdot) + \xi(\cdot)$, so that $\mathbf{E}(\lambda(\cdot)) = 0$. It is the structure of the process $v(\cdot)$ in terms of a linear combination of a fixed number of spatial basis functions that defines the SRE model for $\lambda(\cdot)$:

$$\lambda(\mathbf{s}) = \sum_{l=1}^r \phi_l(\mathbf{s}) \eta_l + \xi(\mathbf{s}); \quad \mathbf{s} \in D,$$

where $\boldsymbol{\eta} \equiv (\eta_1, \dots, \eta_r)^\top$ is an r -variate random vector, and $\boldsymbol{\phi}(\cdot) \equiv (\phi_1(\cdot), \dots, \phi_r(\cdot))^\top$ is an r -dimensional vector of pre-specified spatial basis functions. Sometimes, $\boldsymbol{\phi}(\cdot)$ contains basis functions of multiple resolutions (e.g., wavelets), they may or may not be orthogonal, and they may or may not have compact support. The basis functions chosen should be able to adequately reconstruct realizations of $Y(\cdot)$; an empirical spectral-based approach that can ensure this is discussed in [Zammit-Mangion, Sanguinetti, and Kadirkamanathan \(2012\)](#).

In order to cater for different observation supports $\{B_j\}$ (defined below), it is convenient to assume a discretized domain of interest $D^G \equiv \{A_i \subset D : i = 1, \dots, N\}$ that is made up of N small, non-overlapping basic areal units or BAUs ([Nguyen *et al.* 2012](#)), and $D = \bigcup_{i=1}^N A_i$. The set D^G of BAUs is a discretization, or ‘tiling,’ of the original domain D , and typically $N \gg r$. The process $\{Y(\mathbf{s}) : \mathbf{s} \in D\}$ is then averaged over the BAUs, giving the vector $\mathbf{Y} = (Y_i : i = 1, \dots, N)^\top$, where

$$Y_i \equiv \frac{1}{|A_i|} \int_{A_i} Y(\mathbf{s}) d\mathbf{s}; \quad i = 1, \dots, N, \quad (1)$$

and N is the number of BAUs. At this BAU level,

$$Y_i = \mathbf{t}_i^\top \boldsymbol{\alpha} + v_i + \xi_i, \quad (2)$$

where for $i = 1, \dots, N$, $\mathbf{t}_i \equiv \frac{1}{|A_i|} \int_{A_i} \mathbf{t}(\mathbf{s}) d\mathbf{s}$, $v_i \equiv \frac{1}{|A_i|} \int_{A_i} v(\mathbf{s}) d\mathbf{s}$, and ξ_i is specified below. The SRE model specifies that the small-scale random variation is $v(\cdot) = \boldsymbol{\phi}(\cdot)^\top \boldsymbol{\eta}$, and hence in terms of the discretization onto D^G ,

$$v_i = \left(\frac{1}{|A_i|} \int_{A_i} \boldsymbol{\phi}(\mathbf{s}) d\mathbf{s} \right)^\top \boldsymbol{\eta}; \quad i = 1, \dots, N,$$

so that $\mathbf{v} = \mathbf{S}\boldsymbol{\eta}$, where \mathbf{S} is the $N \times r$ matrix defined as follows:

$$\mathbf{S} \equiv \left(\frac{1}{|A_i|} \int_{A_i} \boldsymbol{\phi}(\mathbf{s}) d\mathbf{s} : i = 1, \dots, N \right)^\top. \quad (3)$$

In **FRK**, we assume that $\boldsymbol{\eta}$ is an r -dimensional Gaussian vector with mean zero and $r \times r$ covariance matrix \mathbf{K} , and estimation of \mathbf{K} is based on likelihood methods; we denote this variant of FRK as FRK-V (where recall that ‘V’ stands for ‘vanilla’). If some structure is imposed on $\text{VAR}(\boldsymbol{\eta})$ in terms of parameters $\boldsymbol{\vartheta}$, then $\mathbf{K} = \mathbf{K}_o(\boldsymbol{\vartheta})$ and $\boldsymbol{\vartheta}$ needs to be estimated;

we denote this variant as FRK-M (where recall that ‘M’ stands for ‘model’). Frequently, the resolution of the BAUs is sufficiently fine, and the basis functions are sufficiently smooth, so that \mathbf{S} can be approximated:

$$\mathbf{S} \approx (\phi(\mathbf{s}_i) : i = 1, \dots, N)^\top, \quad (4)$$

where $\{\mathbf{s}_i : i = 1, \dots, N\}$ are the centroids of the BAUs. Since small BAUs are always assumed, this approximation is used throughout **FRK**.

In **FRK**, we do not directly model $\xi(\mathbf{s})$, since we are only interested in its discretized version. Rather, we assume that $\xi_i \equiv \frac{1}{|A_i|} \int_{A_i} \xi(\mathbf{s}) d\mathbf{s}$ has a Gaussian distribution with mean zero and variance

$$\text{VAR}(\xi_i) = \sigma_\xi^2 v_{\xi,i},$$

where σ_ξ^2 is a parameter to be estimated, and the weights $\{v_{\xi,1}, \dots, v_{\xi,N}\}$ are known and represent heteroscedasticity. These weights are typically generated from domain knowledge; they may, for example, correspond to topographical features such as terrain roughness (Zammit-Mangion, Rougier, Schön, Lindgren, and Bamber 2015). Since we specified $\xi(\cdot)$ to be ‘almost’ spatially uncorrelated, it is reasonable to assume that the variables representing the discretized fine-scale variation, $\{\xi_i : i = 1, \dots, N\}$, are uncorrelated. From (2), we can write

$$\mathbf{Y} = \mathbf{T}\boldsymbol{\alpha} + \mathbf{S}\boldsymbol{\eta} + \boldsymbol{\xi}, \quad (5)$$

where $\mathbf{T} \equiv (\mathbf{t}_i : i = 1, \dots, N)^\top$, $\boldsymbol{\xi} \equiv (\xi_i : i = 1, \dots, N)^\top$, and $\text{VAR}(\boldsymbol{\xi}) \equiv \sigma_\xi^2 \mathbf{V}_\xi$, for known $\mathbf{V}_\xi \equiv \text{diag}(v_{\xi,1}, \dots, v_{\xi,N})$.

We now assume that the hidden (or latent) process, $Y(\cdot)$, is observed with m footprints (possibly overlapping) spanning one or more BAUs, where typically $m \gg r$ (note that both $m > N$ and $N \geq m$ are possible). We thus define the observation domain as $D^O \equiv \{\cup_{i \in c_j} A_i : j = 1, \dots, m\}$, where c_j is a non-empty set in $2^{\{1, \dots, N\}}$, the power set of $\{1, \dots, N\}$, and $m = |D^O|$. For illustration, consider the simple case of the discretized domain being made up of three BAUs. Then $D^G = \{A_1, A_2, A_3\}$ and, for example, $D^O = \{B_1, B_2\}$, where $B_1 = A_1 \cup A_2$ (i.e., $c_1 = \{1, 2\}$) and $B_2 = A_3$ (i.e., $c_2 = \{3\}$). Catering for different footprints is important for remote sensing applications in which satellite-instrument footprints can widely differ (e.g., Zammit-Mangion *et al.* 2015).

Each $B_j \in D^O$ is either a BAU or a union of BAUs. Measurement of \mathbf{Y} is imperfect: We define the measurement process as noisy measurements of the process averaged over the footprints

$$Z_j \equiv Z(B_j) = \left(\frac{\sum_{i=1}^N Y_i w_{ij}}{\sum_{i=1}^N w_{ij}} \right) + \left(\frac{\sum_{i=1}^N \delta_i w_{ij}}{\sum_{i=1}^N w_{ij}} \right) + \epsilon_j; \quad B_j \in D^O, \quad (6)$$

where the weights,

$$w_{ij} = |A_i| \mathbb{I}(A_i \subset B_j); \quad i = 1, \dots, N; \quad j = 1, \dots, m; \quad B_j \in D^O,$$

depend on the areas of the BAUs, and $\mathbb{I}(\cdot)$ is the indicator function. Currently, in **FRK**, BAUs of equal area are assumed, but we give (6) in its most general form. The random quantities $\{\delta_i\}$ and $\{\epsilon_i\}$ capture the imperfections of the measurement. Better known is the measurement-error component ϵ_i , which is assumed to be mean-zero Gaussian distributed. The component δ_i captures any bias in the measurement at the BAU level, which has the

interpretation of an intra-BAU systematic error. These systematic errors are BAU-specific, that is, the $\{\delta_i\}$ are uncorrelated with mean zero and variance

$$\text{VAR}(\delta_i) = \sigma_\delta^2 v_{\delta,i},$$

where σ_δ^2 is a parameter to be estimated, and $\{v_{\delta,1}, \dots, v_{\delta,N}\}$ represent known heteroscedasticity.

We assume that \mathbf{Y} and $\boldsymbol{\delta}$ are independent. We also assume that the observations are conditionally independent, when conditioned on \mathbf{Y} and $\boldsymbol{\delta}$. Equivalently, we assume that the measurement errors $\{\epsilon_j : j = 1, \dots, m\}$ are independent with $\text{VAR}(\epsilon_j) = \sigma_\epsilon^2 v_{\epsilon,j}$.

We represent the data as $\mathbf{Z} \equiv (Z_j : j = 1, \dots, m)^\top$. Then, since each element in D^O is the union of subsets of D^G , one can construct a matrix

$$\mathbf{C}_Z \equiv \left(\frac{w_{ij}}{\sum_{l=1}^N w_{lj}} : i = 1, \dots, N; j = 1, \dots, m \right),$$

such that

$$\mathbf{Z} = \mathbf{C}_Z \mathbf{Y} + \mathbf{C}_Z \boldsymbol{\delta} + \boldsymbol{\epsilon},$$

where the three components are independent, $\boldsymbol{\epsilon} \equiv (\epsilon_j : j = 1, \dots, m)^\top$, and $\text{VAR}(\boldsymbol{\epsilon}) = \boldsymbol{\Sigma}_\epsilon \equiv \sigma_\epsilon^2 \mathbf{V}_\epsilon \equiv \sigma_\epsilon^2 \text{diag}(v_{\epsilon,1}, \dots, v_{\epsilon,m})$ is an $m \times m$ diagonal covariance matrix. The matrix $\boldsymbol{\Sigma}_\epsilon$ is assumed known from the properties of the measurement. If it is not known, \mathbf{V}_ϵ is fixed to \mathbf{I} and σ_ϵ^2 is estimated using variogram techniques (Kang, Liu, and Cressie 2009). Notice that the rows of the matrix \mathbf{C}_Z sum to 1.

It will be convenient to re-write

$$\mathbf{Z} = \mathbf{T}_Z \boldsymbol{\alpha} + \mathbf{S}_Z \boldsymbol{\eta} + \boldsymbol{\xi}_Z + \boldsymbol{\delta}_Z + \boldsymbol{\epsilon}, \quad (7)$$

where $\mathbf{T}_Z \equiv \mathbf{C}_Z \mathbf{T}$, $\mathbf{S}_Z \equiv \mathbf{C}_Z \mathbf{S}$, $\boldsymbol{\xi}_Z \equiv \mathbf{C}_Z \boldsymbol{\xi}$, $\boldsymbol{\delta}_Z \equiv \mathbf{C}_Z \boldsymbol{\delta}$, $\text{VAR}(\boldsymbol{\xi}_Z) = \sigma_\xi^2 \mathbf{V}_{\xi,Z} \equiv \sigma_\xi^2 \mathbf{C}_Z \mathbf{V}_\xi \mathbf{C}_Z^\top$, $\text{VAR}(\boldsymbol{\delta}_Z) = \sigma_\delta^2 \mathbf{V}_{\delta,Z} \equiv \sigma_\delta^2 \mathbf{C}_Z \mathbf{V}_\delta \mathbf{C}_Z^\top$, and where $\mathbf{V}_\delta \equiv \text{diag}(v_{\delta,1}, \dots, v_{\delta,N})$ is known. Then, recalling that $\mathbf{E}(\boldsymbol{\eta}) = \mathbf{0}$ and $\mathbf{E}(\boldsymbol{\xi}_Z) = \mathbf{E}(\boldsymbol{\delta}_Z) = \mathbf{E}(\boldsymbol{\epsilon}) = \mathbf{0}$,

$$\begin{aligned} \mathbf{E}(\mathbf{Z}) &= \mathbf{T}_Z \boldsymbol{\alpha}, \\ \text{VAR}(\mathbf{Z}) &= \mathbf{S}_Z \mathbf{K} \mathbf{S}_Z^\top + \sigma_\xi^2 \mathbf{C}_Z \mathbf{V}_\xi \mathbf{C}_Z^\top + \sigma_\delta^2 \mathbf{C}_Z \mathbf{V}_\delta \mathbf{C}_Z^\top + \sigma_\epsilon^2 \mathbf{V}_\epsilon. \end{aligned}$$

In practice, it is not always possible for each B_j to include entire BAUs. For simplicity, in **FRK** we assume that the observation footprint overlaps a BAU if and only if the BAU centroid lies within the footprint. Frequently, point-referenced data are included in \mathbf{Z} . In this case, each data point is attributed to a specific BAU and it is possible to have multiple observations of the process defined on the same BAU.

We collect the unknown parameters in the set $\boldsymbol{\theta} \equiv \{\boldsymbol{\alpha}, \sigma_\xi^2, \sigma_\delta^2, \mathbf{K}\}$ for FRK-V and $\boldsymbol{\theta}_o \equiv \{\boldsymbol{\alpha}, \sigma_\xi^2, \sigma_\delta^2, \boldsymbol{\vartheta}\}$ for FRK-M for which $\mathbf{K} = \mathbf{K}_o(\boldsymbol{\vartheta})$; their estimation is the subject of Section 2.2. If the parameters in $\boldsymbol{\theta}$ or $\boldsymbol{\theta}_o$ are known, an inversion that uses the Sherman-Woodbury identity (Henderson and Searle 1981) allows spatial prediction at any BAU in D^G . Estimates of $\boldsymbol{\theta}$ are substituted into these spatial predictors to yield FRK-V. Similarly, estimates of $\boldsymbol{\theta}_o$ substituted into the spatial-prediction equations yield FRK-M.

In **FRK**, we allow the prediction set D^P to be as flexible as D^O ; specifically, $D^P \subset \{\cup_{i \in \tilde{c}_k} A_i : k = 1, \dots, N_P\}$, where \tilde{c}_k is a non-empty set in $2^{\{1, \dots, N\}}$ and N_P is the number of prediction

areas. We can thus predict both at the individual BAU level or averages over an area spanning multiple BAUs, and these prediction regions may overlap. This is an important change-of-support feature of **FRK**. We provide the FRK equations in Section 2.3.

2.2. Parameter estimation using an EM algorithm

In all its generality, parameter estimation with the model of Section 2.1 is problematic due to confounding between δ and ξ . In **FRK**, the user thus needs to choose between modeling the intra-BAU systematic errors (in which case σ_ξ^2 is fixed to 0) or the process' fine-scale variation (in which case σ_δ^2 is fixed to 0). We describe below the estimation procedure for the latter case; due to symmetry, the estimation equations of the former case can be simply obtained by replacing the subscript ξ with δ . However, which case is chosen by the user has a considerable impact on the prediction equations for Y (Section 2.3). Recall that the measurement-error covariance matrix Σ_ϵ is assumed known from measurement characteristics, or estimated using variogram techniques prior to estimating the remaining parameters described below. For conciseness, in this section we use θ to denote the parameters in both FRK-V and FRK-M, only distinguishing when necessary.

We carry out parameter estimation using an expectation-maximization (EM) algorithm (similar to [Katzfuss and Cressie 2011](#); [Nguyen et al. 2014](#)) with (7) as our model. Define the complete-data likelihood $L_c(\theta) \equiv [\eta, \mathbf{Z} \mid \theta]$ (with ξ_Z integrated out), where $[\cdot]$ denotes the probability distribution of its argument. The EM algorithm proceeds by first computing the conditional expectation (conditional on the data) of the complete-data log-likelihood at the current parameter estimates (the E-step) and, second, maximizing this function with respect to the parameters (the M-step). In mathematical notation, in the E-step the function

$$Q(\theta \mid \theta^{(l)}) \equiv E(\ln L_c(\theta) \mid \mathbf{Z}, \theta^{(l)}),$$

is found for some current estimate $\theta^{(l)}$. In the M-step, the updated parameter estimates

$$\theta^{(l+1)} = \arg \max_{\theta} Q(\theta \mid \theta^{(l)}),$$

are found.

The E-step boils down to finding the conditional distribution of η at the current parameter estimates. One can use standard results in Gaussian conditioning (e.g., [Rasmussen and Williams 2006](#), Appendix A) to show from the joint distribution, $[\eta, \mathbf{Z} \mid \theta^{(l)}]$, that

$$\eta \mid \mathbf{Z}, \theta^{(l)} \sim \text{Gau}(\mu_\eta^{(l)}, \Sigma_\eta^{(l)}),$$

where

$$\begin{aligned} \mu_\eta^{(l)} &= \Sigma_\eta^{(l)} \mathbf{S}_Z^\top (\mathbf{D}_Z^{(l)})^{-1} (\mathbf{Z} - \mathbf{T}_Z \alpha^{(l)}), \\ \Sigma_\eta^{(l)} &= \left(\mathbf{S}_Z^\top (\mathbf{D}_Z^{(l)})^{-1} \mathbf{S}_Z + (\mathbf{K}^{(l)})^{-1} \right)^{-1}, \end{aligned}$$

where $\mathbf{D}_Z^{(l)} \equiv (\sigma_\xi^2)^{(l)} \mathbf{V}_{\xi, Z} + \Sigma_\epsilon$, and where $\mathbf{K}^{(l)}$ is defined below.

The update for α is

$$\alpha^{(l+1)} = \left(\mathbf{T}_Z^\top (\mathbf{D}_Z^{(l+1)})^{-1} \mathbf{T}_Z \right)^{-1} \mathbf{T}_Z^\top (\mathbf{D}_Z^{(l+1)})^{-1} (\mathbf{Z} - \mathbf{S}_Z \mu_\eta^{(l)}). \quad (8)$$

In FRK-V, the update for $\mathbf{K}^{(l+1)}$ is

$$\mathbf{K}^{(l+1)} = \boldsymbol{\Sigma}_\eta^{(l)} + \boldsymbol{\mu}_\eta^{(l)} \boldsymbol{\mu}_\eta^{(l)\top},$$

while in FRK-M, where recall that $\mathbf{K} = \mathbf{K}_o(\boldsymbol{\vartheta})$, the update is

$$\boldsymbol{\vartheta}^{(l+1)} = \arg \max_{\boldsymbol{\vartheta}} \ln \left| \mathbf{K}_o(\boldsymbol{\vartheta})^{-1} \right| - \text{tr} \left(\mathbf{K}_o(\boldsymbol{\vartheta})^{-1} \left(\boldsymbol{\Sigma}_\eta^{(l)} + \boldsymbol{\mu}_\eta^{(l)} \boldsymbol{\mu}_\eta^{(l)\top} \right) \right),$$

which is numerically optimized using the function `optim` with $\boldsymbol{\vartheta}^{(l)}$ as the initial vector.

The update for σ_ξ^2 requires the solution to

$$\begin{aligned} \text{tr}((\boldsymbol{\Sigma}_\epsilon + (\sigma_\xi^2)^{(l+1)} \mathbf{V}_{\xi,Z})^{-1} \mathbf{V}_{\xi,Z}) = \\ \text{tr}((\boldsymbol{\Sigma}_\epsilon + (\sigma_\xi^2)^{(l+1)} \mathbf{V}_{\xi,Z})^{-1} \mathbf{V}_{\xi,Z} (\boldsymbol{\Sigma}_\epsilon + (\sigma_\xi^2)^{(l+1)} \mathbf{V}_{\xi,Z})^{-1} \boldsymbol{\Omega}), \end{aligned} \quad (9)$$

where

$$\begin{aligned} \boldsymbol{\Omega} \equiv \mathbf{S}_Z \boldsymbol{\Sigma}_\eta^{(l)} \mathbf{S}_Z^\top + \mathbf{S}_Z \boldsymbol{\mu}_\eta^{(l)} \boldsymbol{\mu}_\eta^{(l)\top} \mathbf{S}_Z^\top - 2 \mathbf{S}_Z \boldsymbol{\mu}_\eta^{(l)} (\mathbf{Z} - \mathbf{T}_Z \boldsymbol{\alpha}^{(l+1)})^\top + \\ (\mathbf{Z} - \mathbf{T}_Z \boldsymbol{\alpha}^{(l+1)}) (\mathbf{Z} - \mathbf{T}_Z \boldsymbol{\alpha}^{(l+1)})^\top. \end{aligned} \quad (10)$$

The solution to (9), namely $(\sigma_\xi^2)^{(l+1)}$, is found numerically using `uniroot` after (8) is substituted into (10). Then $\boldsymbol{\alpha}^{(l+1)}$ is found by substituting $(\sigma_\xi^2)^{(l+1)}$ into (8). Computational simplifications are possible when $\mathbf{V}_{\xi,Z}$ and $\boldsymbol{\Sigma}_\epsilon$ are diagonal, since then only the diagonal of $\boldsymbol{\Omega}$ needs to be computed. Further simplifications are possible when $\mathbf{V}_{\xi,Z}$ and $\boldsymbol{\Sigma}_\epsilon$ are proportional to the identity matrix, with constants of proportionality γ_1 and γ_2 , respectively. In this case,

$$(\sigma_\xi^2)^{(l+1)} = \frac{1}{\gamma_1} \left(\frac{\text{tr}(\boldsymbol{\Omega})}{m} - \gamma_2 \right),$$

where recall that m is the dimension of the data vector \mathbf{Z} and $\boldsymbol{\alpha}^{(l+1)}$ is, in this special case, the ordinary-least-squares estimate given $\boldsymbol{\mu}_\eta^{(l)}$ (see (8)). These simplifications are used by **FRK** whenever possible.

Convergence of the EM algorithm is assessed using the (incomplete-data) log-likelihood function at each iteration,

$$\ln [\mathbf{Z} \mid \boldsymbol{\alpha}^{(l)}, \mathbf{K}^{(l)}, (\sigma_\xi^2)^{(l)}] = -\frac{m}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_Z^{(l)}| - \frac{1}{2} (\mathbf{Z} - \mathbf{T}_Z \boldsymbol{\alpha}^{(l)})^\top (\boldsymbol{\Sigma}_Z^{(l)})^{-1} (\mathbf{Z} - \mathbf{T}_Z \boldsymbol{\alpha}^{(l)}),$$

where

$$\boldsymbol{\Sigma}_Z^{(l)} = \mathbf{S}_Z \mathbf{K}^{(l)} \mathbf{S}_Z^\top + \mathbf{D}_Z^{(l)},$$

and recall that $\mathbf{D}_Z^{(l)} \equiv (\sigma_\xi^2)^{(l)} \mathbf{V}_{\xi,Z} + \boldsymbol{\Sigma}_\epsilon$. Efficient computation of the log-likelihood is facilitated through the use of the Sherman-Morrison-Woodbury matrix identity and a matrix-determinant lemma (e.g., [Henderson and Searle 1981](#)). Specifically, the operations

$$\begin{aligned} (\boldsymbol{\Sigma}_Z^{(l)})^{-1} &= (\mathbf{D}_Z^{(l)})^{-1} - (\mathbf{D}_Z^{(l)})^{-1} \mathbf{S}_Z \left[(\mathbf{K}^{(l)})^{-1} + \mathbf{S}_Z^\top (\mathbf{D}_Z^{(l)})^{-1} \mathbf{S}_Z \right]^{-1} \mathbf{S}_Z^\top (\mathbf{D}_Z^{(l)})^{-1}, \\ |\boldsymbol{\Sigma}_Z^{(l)}| &= \left| (\mathbf{K}^{(l)})^{-1} + \mathbf{S}_Z^\top (\mathbf{D}_Z^{(l)})^{-1} \mathbf{S}_Z \right| |\mathbf{K}^{(l)}| |\mathbf{D}_Z^{(l)}|, \end{aligned}$$

ensure that we only deal with vectors of length m and matrices of size $r \times r$, where typically the fixed rank $r \ll m$, the dataset size.

2.3. Prediction

The prediction task is to make inference on the hidden Y -process over a set of prediction regions D^P . Consider the process $\{Y_P(\tilde{B}_k) : k = 1, \dots, N_P\}$, which is derived from the Y process and, similar to the observations, is constructed using the BAUs $\{A_i : i = 1, \dots, N\}$. Here, N_P is the number of areas at which spatial prediction takes place, and is equal to $|D^P|$. Then,

$$Y_{P,k} \equiv Y_P(\tilde{B}_k) = \left(\frac{\sum_{i=1}^N Y_i \tilde{w}_{ik}}{\sum_{i=1}^N \tilde{w}_{ik}} \right); \quad \tilde{B}_k \in D^P,$$

where the weights are

$$\tilde{w}_{ik} = |A_i| \mathbb{I}(A_i \subset \tilde{B}_k); \quad i = 1, \dots, N; \quad k = 1, \dots, N_P; \quad \tilde{B}_k \in D^P.$$

Define $\mathbf{Y}_P \equiv (Y_{P,k} : k = 1, \dots, N_P)^\top$. Then, since each element in D^P is the union of subsets of D^G , one can construct a matrix,

$$\mathbf{C}_P \equiv \left(\frac{\tilde{w}_{ik}}{\sum_{l=1}^N \tilde{w}_{lk}} : i = 1, \dots, N; k = 1, \dots, N_P \right), \quad (11)$$

the rows of which sum to 1, such that

$$\mathbf{Y}_P = \mathbf{C}_P \mathbf{Y} = \mathbf{T}_P \boldsymbol{\alpha} + \mathbf{S}_P \boldsymbol{\eta} + \boldsymbol{\xi}_P,$$

where $\mathbf{T}_P \equiv \mathbf{C}_P \mathbf{T}$, $\mathbf{S}_P \equiv \mathbf{C}_P \mathbf{S}$, $\boldsymbol{\xi}_P \equiv \mathbf{C}_P \boldsymbol{\xi}$ and $\text{VAR}(\boldsymbol{\xi}_P) = \sigma_\xi^2 \mathbf{V}_{\xi,P} \equiv \sigma_\xi^2 \mathbf{C}_P \mathbf{V}_\xi \mathbf{C}_P^\top$. As with the observations, the prediction regions $\{\tilde{B}_k\}$ may overlap. In practice, it may not always be possible for each \tilde{B}_k to include entire BAUs. In this case, we assume that a prediction region contains a BAU if and only if the BAU centroid lies within the region.

Let l^* denote the EM iteration number at which convergence is deemed to have been reached. The final estimates are then

$$\hat{\boldsymbol{\mu}}_\eta \equiv \boldsymbol{\mu}_\eta^{(l^*)}, \quad \hat{\boldsymbol{\Sigma}}_\eta \equiv \boldsymbol{\Sigma}_\eta^{(l^*)}, \quad \hat{\boldsymbol{\alpha}} \equiv \boldsymbol{\alpha}^{(l^*)}, \quad \hat{\mathbf{K}} \equiv \mathbf{K}^{(l^*)}, \quad \hat{\sigma}_\xi^2 \equiv (\sigma_\xi^2)^{(l^*)}, \quad \text{and} \quad \hat{\sigma}_\delta^2 \equiv (\sigma_\delta^2)^{(l^*)}.$$

Recall from Section 2.2 that the user needs to attribute fine-scale variation at the BAU level to either the measurement process or the hidden process Y . This leads to the following two cases.

Case 1: $\sigma_\xi^2 = 0$ and estimate σ_δ^2 . The prediction vector $\hat{\mathbf{Y}}_P$ and covariance matrix $\boldsymbol{\Sigma}_{Y_P|Z}$, corresponding to the first two moments from the predictive distribution $[\mathbf{Y}_P | \mathbf{Z}]$ when $\sigma_\xi^2 = 0$, are

$$\begin{aligned} \hat{\mathbf{Y}}_P &\equiv \mathbb{E}(\mathbf{Y}_P | \mathbf{Z}) = \mathbf{T}_P \hat{\boldsymbol{\alpha}} + \mathbf{S}_P \hat{\boldsymbol{\mu}}_\eta, \\ \boldsymbol{\Sigma}_{Y_P|Z} &\equiv \text{VAR}(\mathbf{Y}_P | \mathbf{Z}) = \mathbf{S}_P \hat{\boldsymbol{\Sigma}}_\eta \mathbf{S}_P^\top. \end{aligned}$$

Under the assumptions taken, $[\mathbf{Y}_P | \mathbf{Z}]$ is a $\text{Gau}(\hat{\mathbf{Y}}_P, \boldsymbol{\Sigma}_{Y_P|Z})$ distribution. Note that all calculations are made after substituting in the EM-estimated parameters, and that $\hat{\sigma}_\delta^2$ is present in the estimated parameters.

Case 2: $\sigma_\delta^2 = 0$ and estimate σ_ξ^2 (default). To cater for arbitrary observation and prediction support, we predict \mathbf{Y}_P by first carrying out prediction over the full vector \mathbf{Y} , that is, at the BAU level, and then transforming linearly to obtain $\hat{\mathbf{Y}}_P$ through the use of the matrix \mathbf{C}_P . It is easy to see that if $\hat{\mathbf{Y}}$ is an optimal (squared-error-loss matrix criterion) predictor of \mathbf{Y} , then $\mathbf{A}\hat{\mathbf{Y}}$ is an optimal predictor of $\mathbf{A}\mathbf{Y}$, where \mathbf{A} is any matrix with N columns.

Let $\mathbf{W} \equiv (\boldsymbol{\eta}^\top, \boldsymbol{\xi}^\top)^\top$ and $\boldsymbol{\Pi} \equiv (\mathbf{S}, \mathbf{I})$. Then (5) can be re-written as $\mathbf{Y} = \mathbf{T}\boldsymbol{\alpha} + \boldsymbol{\Pi}\mathbf{W}$, and

$$\begin{aligned}\hat{\mathbf{Y}} &\equiv \mathbf{E}(\mathbf{Y} \mid \mathbf{Z}) = \mathbf{T}\hat{\boldsymbol{\alpha}} + \boldsymbol{\Pi}\hat{\mathbf{W}}, \\ \boldsymbol{\Sigma}_{Y|Z} &\equiv \text{VAR}(\mathbf{Y} \mid \mathbf{Z}) = \boldsymbol{\Pi}\boldsymbol{\Sigma}_W\boldsymbol{\Pi}^\top,\end{aligned}\tag{12}$$

for

$$\begin{aligned}\hat{\mathbf{W}} &\equiv \boldsymbol{\Sigma}_W\boldsymbol{\Pi}^\top\mathbf{C}_Z^\top\boldsymbol{\Sigma}_\epsilon^{-1}(\mathbf{Z} - \mathbf{T}_Z\hat{\boldsymbol{\alpha}}), \\ \boldsymbol{\Sigma}_W &\equiv \left(\boldsymbol{\Pi}^\top\mathbf{C}_Z^\top\boldsymbol{\Sigma}_\epsilon^{-1}\mathbf{C}_Z\boldsymbol{\Pi} + \boldsymbol{\Lambda}^{-1}\right)^{-1},\end{aligned}$$

and the block-diagonal matrix $\boldsymbol{\Lambda} \equiv \text{bdiag}(\hat{\mathbf{K}}, \hat{\sigma}_\xi^2\mathbf{V}_\xi)$, where $\text{bdiag}(\cdot)$ returns a block diagonal matrix of its matrix arguments. Note that all calculations are made after substituting in the EM-estimated parameters.

For both Cases 1 and 2 it follows that $\hat{\mathbf{Y}}_P = \mathbf{E}(\mathbf{Y}_P \mid \mathbf{Z}) = \mathbf{C}_P\hat{\mathbf{Y}}$ and

$$\boldsymbol{\Sigma}_{Y_P|Z} = \text{VAR}(\mathbf{Y}_P \mid \mathbf{Z}) = \mathbf{C}_P\boldsymbol{\Sigma}_{Y|Z}\mathbf{C}_P^\top.\tag{13}$$

Note that for Case 2 we need to obtain predictions for $\boldsymbol{\xi}_P$ which, unlike those for $\boldsymbol{\eta}$, are not a by-product of the EM algorithm of Section 2.2. Sparse-matrix operations (Zammit-Mangion and Rougier 2018) are used to facilitate the computation of (13) when possible.

3. FRK package structure and usage

In this section we discuss the layout and the interface of the package, and we show its use on the **meuse** dataset under ‘simple usage’ and ‘advanced usage.’ The former attempts to construct the SRE model automatically from characteristics of the data, while the latter gives the user more control through use of additional commands. The **meuse** dataset is not large and contains 155 readings of heavy-metal abundance in a region of The Netherlands along the river Meuse. For more details on the dataset see the vignette titled ‘gstat’ in the package **gstat** (Pebesma 2004).

3.1. Usage overview

By leveraging the flexibility of the spatial and spatio-temporal objects in the **sp** (Bivand, Pebesma, and Gómez-Rubio 2013) and **spacetime** (Pebesma 2012) packages, **FRK** provides a consistent, easy-to-use interface for the user, irrespective of whether the datasets have different spatial supports, irrespective of the manifold being used, irrespective of whether or not a temporal dimension needs to be included, and irrespective of the ‘prediction resolution.’ In Figure 1 we provide a partial unified modeling language (UML) diagram summarizing the important package classes and their interaction with the packages **sp** and **spacetime**, while in Table 1 we provide a brief summary of these classes. BAUs should be ‘Spatial’ or ‘ST’ pixel

Class	Description
<code>'Basis'</code>	Defines basis functions on a specified manifold.
<code>'Basis_obj'</code>	A virtual class that other basis classes inherit from.
<code>'manifold'</code>	A virtual class that other manifold classes inherit from.
<code>'measure'</code>	Defines objects that compute distances on a specified manifold.
<code>'plane'</code> , <code>'real_line'</code> , <code>'sphere'</code> , <code>'STplane'</code> , <code>'STsphere'</code> , <code>'STmanifold'</code>	Subclasses that inherit from the virtual class <code>'manifold'</code> .
<code>'SRE'</code>	Defines the spatial-random-effects model, which is used to do FRK.
<code>'TensorP_Basis'</code>	Tensor product of two basis functions.

Table 1: Important class definitions in **FRK**.

or polygon objects, while the data can also be point objects (although they are subsequently mapped to BAUs by **FRK**). Each `'Spatial'` and `'ST'` object is equipped with a coordinate reference system (CRS), which needs to be identical across objects. The main class is the `'SRE'` class, the object of which incorporates all information about fitting and prediction using the data, BAUs, and basis functions.

The basis functions are constructed on a manifold which, at the time of writing, can be \mathbb{R} (`real_line`), \mathbb{R}^2 (`plane`), \mathbb{S}^2 (surface of `sphere`), and their spatio-temporal counterparts (`STplane` and `STsphere`). Some consistency checks are made to ensure that the CRS in the BAUs and the data objects are compatible with the manifold on which the basis functions are constructed. As with `spDists` in the **sp** package, distances on the manifold are either Euclidean or great-circle. The function `spDists` in **sp** is not used, rather a function in an object of class `'measure'` is used for abstraction – this redundant structure is intended to facilitate future implementation of **FRK** on arbitrary manifolds and with arbitrary distance functions. The package **FRK** has support for spatio-temporal data (see Section 6); in this case, basis functions are of class `'TensorP_Basis'` and, as the name implies, are constructed through the tensor product of spatial and temporal basis functions.

The package is built around a straightforward model (outlined in Section 2) and has the capability of handling large datasets (up to a few hundred thousand data points on a standard desktop machine, and a few million on a big memory machine). For linear algebraic calculations, it leverages routines from the **sparseinv** package (Zammit-Mangion 2018, which is built from C code written by Davis 2021) and the R package **Matrix** (Bates and Maechler 2021). The package **INLA** (Lindgren and Rue 2015) is used for finding a non-convex hull of the data points and for placing basis functions irregularly in the domain (if desired).

The user has two levels of control; for simple problems one can call the function `FRK`, in which case basis-function construction and BAU generation is done automatically based on characteristics of the data. Alternatively, for more (advanced) control, the user can follow the following six steps.

- **Step 1:** Place the data into an object with class defined in **sp** or **spacetime**, specifically either `'SpatialPointsDataFrame'` or `'STIDF'` for point-referenced data, and either `'SpatialPolygonsDataFrame'` or `'STFDF'` for polygon-referenced data (Pebesma 2012).

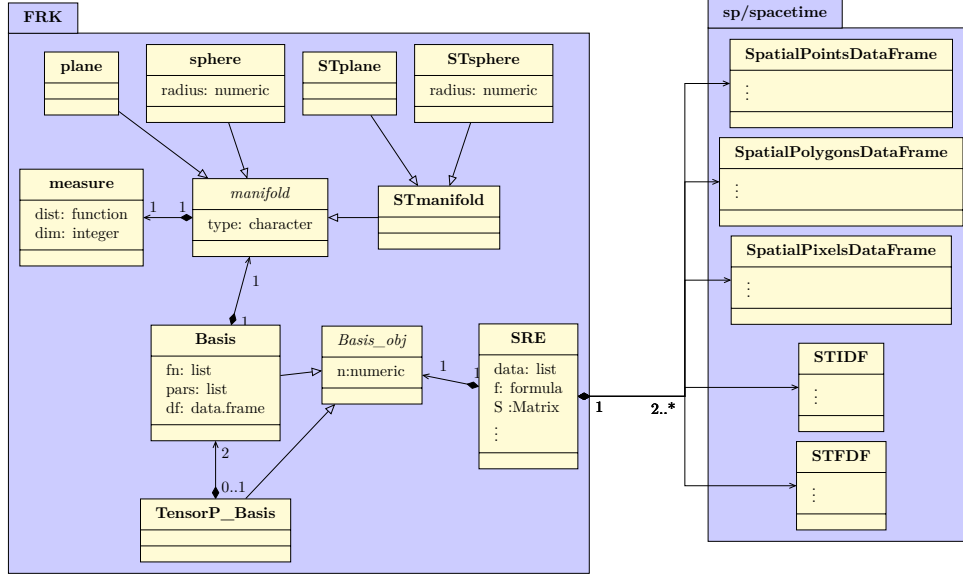


Figure 1: Partial UML diagram showing the most important classes of **FRK** and their interaction to the relevant classes in **sp** and **spacetime**. See Table 1 for a brief description of these classes. For conciseness, in each class diagram (yellow box) only a few attributes are shown and no class operations are listed. Italicized class names indicate virtual classes, an arrow with an open arrowhead indicates inheritance, and a line with a diamond at one end and an arrowhead at another indicates a compositional (“has a”) relationship. The numbers on these lines indicate the number of instances involved in the relationship. For example, the ‘SRE’ class always has two or more **sp** or **spacetime** instances (BAUs and data), while a ‘TensorP_Basis’ object may or may not be needed when setting up the SRE model. In the former case we use the notation ‘2..*’ to denote ‘two or more,’ while in the latter we use ‘0..1’ to note that the user may have 0 or 1 ‘TensorP_Basis’ objects when using **FRK**.

- **Step 2:** Construct a prediction grid of BAUs using `auto_BAUs`, where each BAU is representative of the finest scale upon which we wish to carry out inference (the process is discretized at the BAU level). The BAUs are usually of class ‘SpatialPixelsDataFrame’ for spatial problems (or they could also be of class ‘SpatialPolygonsDataFrame’), and they are of class ‘STFDF’ for spatio-temporal problems.
- **Step 3:** Construct using `auto_basis` a set of regularly or irregularly spaced basis functions. The basis functions can be of various types (e.g., bisquare, Gaussian, or exponential functions).
- **Step 4:** Construct an SRE model using `SRE` from an R formula that identifies the response variable, the covariates, the data, the BAUs, and the basis functions.
- **Step 5:** Estimate the parameters within the SRE model using `SRE.fit`. Estimation is carried out using the EM algorithm described in Section 2.2.
- **Step 6:** Predict either at the BAU level or over arbitrary polygons specified as ‘SpatialPolygon’s or ‘SpatialPolygonDataFrame’s in the spatial case, or as ‘STFDF’s in the spatio-temporal case, using `predict`.

Group	Method/Function	Use
Basis functions	<code>auto_basis</code>	Automatically constructs a set of basis functions on a given manifold based on a supplied dataset.
	<code>local_basis</code>	Manually constructs a set of ‘local’ basis functions from a set of centroids and scale parameters.
	<code>eval_basis</code>	Evaluates basis functions over arbitrary points or polygons.
	<code>remove_basis</code>	Removes basis functions from an object of class ‘Basis’.
	<code>show_basis</code>	Visualizes basis functions.
BAUs	<code>auto_BAUs</code>	Automatically constructs a set of BAUs on a given manifold around a supplied dataset.
Information	<code>BAUs_from_points</code>	Constructs BAUs from point-level data.
	<code>coef</code>	Returns regression coefficients from a fitted SRE model.
	<code>info_fit</code>	Returns information from the EM algorithm (e.g., information on convergence).
	<code>nbasis</code>	Returns the number of basis functions in a ‘Basis’ or ‘SRE’ object.
	<code>nres</code>	Returns the number of basis-function resolutions in a ‘Basis’ or ‘SRE’ object.
FRK operations	<code>opts_FRK\$get</code>	Returns current option settings.
	<code>opts_FRK\$set</code>	Sets an option.
	<code>summary</code>	Returns information on the ‘Basis’ or ‘SRE’ object.
	<code>FRK</code>	Constructs and fits an SRE model from a supplied R formula and dataset.
	<code>predict</code>	Predicts over BAUs or at <code>newdata</code> using a fitted SRE model.
	<code>SRE</code>	Constructs an SRE model from an R formula, data, BAUs, and basis functions.
	<code>SRE.fit</code>	Fits (estimates parameters in) an SRE model.

Table 2: Important methods and functions in **FRK**.

In Table 2 we provide some of the important methods and functions, together with brief descriptions, available to the user of **FRK**.

3.2. Simple usage

In simple cases, the user constructs and fits the SRE model using the function `FRK`, and then prediction is carried out using the function `predict`. The main function `FRK` takes two compulsory arguments: A standard R formula `f` and a list of data objects `data`, and it returns an object of class ‘SRE’. Each of the data objects in the list must be of class ‘SpatialPointsDataFrame’, ‘SpatialPolygonsDataFrame’, ‘STIDF’, or ‘STFDF’, and each must contain the dependent variable defined in `f`. If there are covariates, then the user must supply the covariate data with all the BAUs, that is, at both the BAU measurement locations and at the BAU prediction locations.

The BAUs should be of class ‘`SpatialPolygonsDataFrame`’ or ‘`SpatialPixelsDataFrame`’ (in the spatial case) or ‘`STFDF`’ (in the spatio-temporal case). Note that, unlike conventional spatial modeling tools, covariate information should not be supplied with the data, but with the BAUs. Also note that the intersection of the data support and that of the BAUs should never be null.

When no basis functions or BAUs are supplied, then these are elicited automatically based on characteristics of the supplied dataset(s). The number of basis functions used depends on whether **K** is unstructured or not, on whether the data are spatial only or are spatio-temporal, and on the number of data points. For details, see the package’s manual ([Zammit-Mangion and Sainsbury-Dale 2021](#)). The number of BAUs depends on the domain boundary and on whether the dataset is spatial or spatio-temporal. Domain construction and basis-function placement may make use of geometric functions available in **INLA**. If **INLA** is unavailable, simple geometric methods are used instead.

FRK was not built for small datasets, for which standard exact kriging is fast and memory efficient. However, to illustrate the utility of **FRK**, we consider the `meuse` dataset in the package `sp`. We first consider a simple model with no covariates, in which we model the logarithm of zinc concentrations. Basis functions can either be arranged on a grid by setting `regular = 1` or as a function of data density (using the **INLA** mesher) by setting `regular = 0`.

The `meuse` dataset is first loaded and cast into a ‘`SpatialPointsDataFrame`’.

```
R> library("sp")
R> data("meuse", package = "sp")
R> coordinates(meuse) <- ~x + y
```

Then, **FRK** is invoked as follows.

```
R> library("FRK")
R> f <- log(zinc) ~ 1
R> S <- FRK(f = f, data = list(meuse), regular = 0)
```

The returned ‘`SRE`’ object `S` contains all the information about the fitted SRE model, which can be displayed using the `summary` command.

If we wish to use covariate information, we need to consider BAUs that have covariate information attached to them. Such BAUs are available for this problem in the package `sp` in `meuse.grid`, which we first cast into a ‘`SpatialPixelsDataFrame`’ using the function `gridded` before using them in the SRE model.

```
R> data("meuse.grid", package = "sp")
R> coordinates(meuse.grid) <- ~x + y
R> gridded(meuse.grid) <- TRUE
```

In this example, based on prior exploratory data analysis (see the vignette ‘`gstat`’ in the package **gstat**), we consider the square root of the distance from the centroid of a BAU to the nearest point on the river Meuse as the covariate. Recall that all covariates need to be supplied with the BAUs and not with the data, and **FRK** will throw an error if the data and BAUs have fields in common. In the code below, we first set any common fields to `NULL` in the data object, before running **FRK** using the user-specified BAUs.

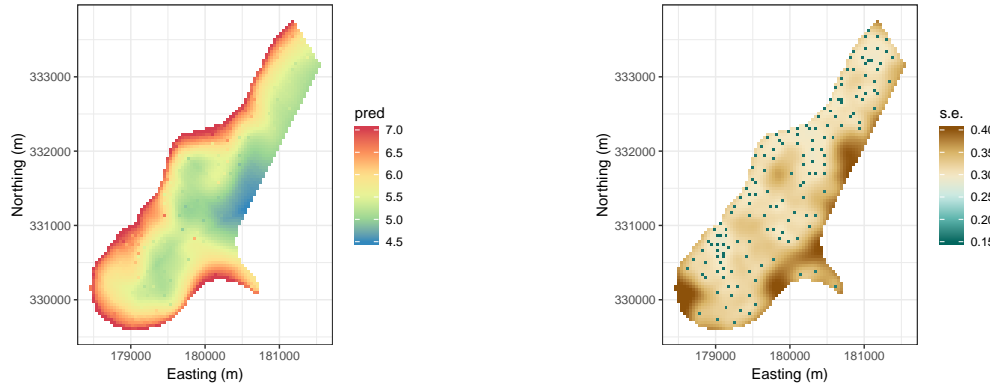


Figure 2: (Left panel) Prediction of log-zinc concentration obtained from **FRK** using the **meuse** dataset. (Right panel) Prediction standard errors for log-zinc concentration. Both quantities are in logs of ppm.

```
R> meuse$soil <- meuse$dist <- meuse$ffreq <- NULL
R> f <- log(zinc) ~ 1 + sqrt(dist)
R> S <- FRK(f = f, data = list(meuse), BAUs = meuse.grid, regular = 0)
```

The other core function, which is also needed for ‘advanced usage,’ is **predict**, which is used to compute prediction and prediction standard errors at all prediction locations. This function takes as compulsory argument the SRE model **S**. If no polygons are specified, prediction is carried out at the BAU level (in space and/or time). An important argument is the flag **obs_fs**, which acts as a choice between Case 1 (process’ fine-scale variance $\sigma_\xi^2 = 0$; **obs_fs** = **TRUE**) and Case 2 (systematic intra-BAU variance $\sigma_\delta^2 = 0$; **obs_fs** = **FALSE**) of Section 2.3.

```
R> Pred <- predict(S, obs_fs = FALSE)
```

The function **Pred** returns the polygons (or, in this case, the BAUs) containing the prediction **mu** and the prediction variance **var**, which can be readily used for visualization. The predictions and prediction standard errors of the model having **sqrt(dist)** as a covariate are depicted in Figure 2. In this instance, Case 2 was used, and the estimate of the fine-scale variance σ_ξ^2 was positive. Hence, the prediction and prediction-error maps exhibit ‘bulls-eye’ features, where the prediction standard errors are much lower in BAUs containing data than in neighboring BAUs not containing data.

Point-level data and predictions

In many cases, the user has one data object or data frame containing both observations and prediction locations with accompanying covariates. Missing observations are then usually denoted as **NA**. Since in **FRK** all covariates are associated with the BAUs and not the data, that one data object needs to be used to construct (i) a second data object where no data are missing and that does not contain missing covariates, and (ii) BAUs at both the observation and prediction locations supplied with their associated covariate data.

For example, assume that the first 10 log-zinc concentrations are missing in the **meuse** dataset.


```
R> data("meuse", package = "sp")
R> meuse[1:10, "zinc"] <- NA
```

Once the data frame is appropriately subsetted, it is then cast as a ‘SpatialPointsDataFrame’ as usual.

```
R> meuse2 <- subset(meuse, !is.na(zinc))
R> meuse2 <- meuse2[, c("x", "y", "zinc")]
R> coordinates(meuse2) <- ~x + y
```

The BAUs, on the other hand, should contain all the data and prediction locations, but not the response variable itself. Their construction is facilitated by the function `BAUs_from_points` which constructs tiny BAUs around the data and prediction locations.

```
R> meuse$zinc <- NULL
R> coordinates(meuse) <- c("x", "y")
R> meuse.grid2 <- BAUs_from_points(meuse)
```

Once BAUs are constructed at both data and prediction locations, FRK may proceed as shown below. Predictions are by default made at both the observed and unobserved BAUs.

```
R> f <- log(zinc) ~ 1 + sqrt(dist)
R> S <- FRK(f = f, data = list(meuse2), BAUs = meuse.grid2, regular = 0)
R> Pred <- predict(S, obs_fs = FALSE)
```

3.3. Advanced usage

The package **FRK** provides several helper functions for facilitating basis-function construction and BAU construction when more control is needed. Harnessing the extra functionality requires following the six steps outlined in Section 3.1.

Step 1: As before, we first load the data and cast it into a ‘SpatialPointsDataFrame’.

```
R> data("meuse", package = "sp")
R> coordinates(meuse) <- ~x + y
```

Step 2: Based on the geometry of the data we now generate BAUs. For this, we use the helper function `auto_BAUs`, which takes several arguments (see `help(auto_BAUs)` for details). In the code below, we instruct the helper function to construct BAUs on the plane, centered around the data in `meuse` with each BAU of size 100×100 m. The `type = "grid"` input indicates that we want a rectangular grid and not a hexagonal lattice (`type = "hex"`) and `convex = -0.05` is a parameter controlling the shape of the domain boundary when `nonconvex_hull = TRUE` (see the help file of `INLA::inla.nonconvex.hull` and [Lindgren and Rue 2015](#) for more details), and the extension of the convex hull of the data when `nonconvex_hull = FALSE` (default).

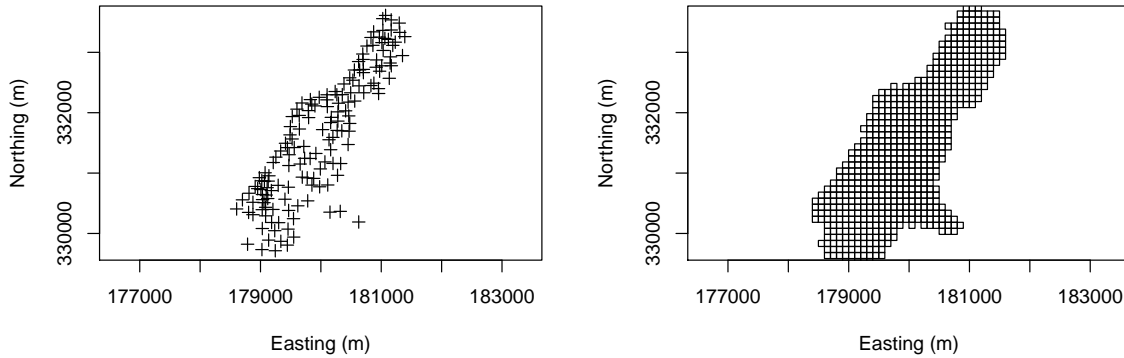


Figure 3: (Left panel) Locations of the `meuse` data. (Right panel) BAUs for FRK with the `meuse` dataset constructed using `auto_BAUs`.

```
R> GridBAUs1 <- auto_BAUs(manifold = plane(), type = "grid",
+   cellsize = c(100, 100), data = meuse, nonconvex_hull = TRUE,
+   convex = -0.05)
```

For the i th BAU, we also need to supply the element $v_{\xi,i}$ (or $v_{\delta,i}$) that describes the heteroscedasticity of the fine-scale variation for that BAU. As described in Section 2.1, this component encompasses all process variation that occurs at the BAU scale and only needs to be known up to a constant of proportionality, σ_{ξ}^2 or σ_{δ}^2 (depending on the chosen model); this constant is estimated using maximum likelihood with `SRE.fit`, which uses the EM algorithm of Section 2.2. Typically, geographic features such as altitude are appropriate, but in this illustration of the package we just set this value to be 1 for all BAUs. This field is labeled `fs`, and `SRE` will throw an error if it is not set.

```
R> GridBAUs1$fs <- 1
```

The data and BAUs are illustrated using the `plot` function in Figure 3. At this stage, the BAUs only contain geographical information. To add covariate information to these BAUs from other ‘Spatial’ objects, the function `sp::over` can be used.

Step 3: FRK decomposes the spatial process as a sum of basis functions that can be constructed using the helper functions `auto_basis` as follows:

```
R> G <- auto_basis(manifold = plane(), data = meuse, regular = 0,
+   nres = 3, type = "bisquare")
```

The argument `nres` indicates the number of basis-function resolutions to use, while `type` indicates the function to use, in this case the bisquare function,

$$b(\mathbf{s}_1, \mathbf{s}_2) \equiv \begin{cases} A\{1 - (\|\mathbf{s}_2 - \mathbf{s}_1\|/r)^2\}^2; & \|\mathbf{s}_2 - \mathbf{s}_1\| \leq r, \\ 0; & \text{otherwise,} \end{cases}$$

where A is the amplitude and r is the aperture. Other options are `"exp"` (the exponential covariance function) and `"Matern32"` (the Matérn covariance function with smoothness parameter equal to 1.5). The basis functions do not need to be positive-definite and users may

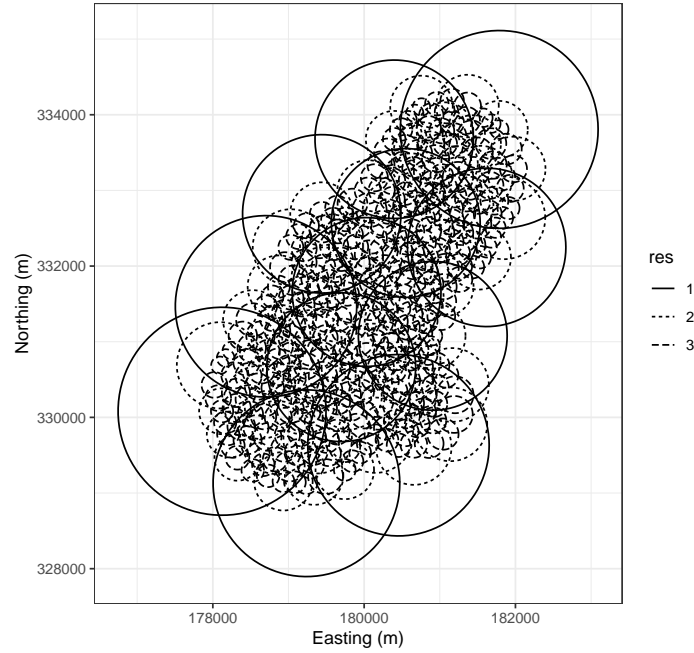


Figure 4: Basis functions automatically generated for the `meuse` dataset with 3 resolutions. The interpretation of the circles changes with the domain and basis-function type. For bisquare functions on the plane, each circle is centered at the basis-function center, and has a radius equal to the function’s aperture. See the help file of `show_basis` for details.

define their own; see Section 5.3. The argument `prune` (not used in this example) may be used to remove basis functions that are not influenced by data: `prune` should be used with care as, in general, basis functions are needed to represent variability in unobserved regions. However, it is useful when implementing FRK-V, where all the columns of \mathbf{S}_Z in (7) need to contain at least one non-zero element in order for $\boldsymbol{\eta}$ to be identifiable. See `help(auto_basis)` for details.

The basis functions can be visualized using `show_basis(G)`; see Figure 4.

Step 4: The SRE model is constructed by supplying an R formula, the data, the BAUs, and the basis functions, to the function `SRE`. If the model contains covariates, one must make sure that they are specified at the BAU-level (and hence attributed to `GridBAUs1`). We use the following formula.

```
R> f <- log(zinc) ~ 1
```

The SRE model is then constructed using the function `SRE`, which essentially bins the data into the BAUs, constructs all the matrices required for estimation, and provides initial guesses for the parameters that need to be estimated. By default, `K_type = "block-exponential"`, which signals the construction of the matrices

$$\mathbf{K}_n(\boldsymbol{\vartheta}) = (\vartheta_{1n} \exp(-d_{ijn}/\vartheta_{2n}) : i, j = 1, \dots, r_n),$$

where d_{ijn} is the distance between the centroids of the i th and j th basis functions at the n th resolution, r_n is the number of basis functions at the n th resolution, $n = 1, \dots, n_{\text{res}}$,

n_{res} is the number of resolutions, ϑ_{1n} is the marginal variance at the n th resolution, and ϑ_{2n} is the e-folding length-scale (i.e., the distance at which the correlation is $1/e$) at the n th resolution. Then the default is $\mathbf{K}_o(\boldsymbol{\vartheta}) = \text{bdiag}(\{\mathbf{K}_n(\boldsymbol{\vartheta}) : n = 1, \dots, n_{\text{res}}\})$, where $\boldsymbol{\vartheta} \equiv (\vartheta_{11}, \vartheta_{21}, \vartheta_{12}, \dots, \vartheta_{2n_{\text{res}}})^\top$ and $\text{bdiag}(\cdot)$ returns a block-diagonal matrix constructed from its arguments.

```
R> S <- SRE(f = f, data = list(meuse), BAUs = GridBAUs1, basis = G,
+   est_error = TRUE, average_in_BAU = FALSE)
```

`K_type = "unstructured"` can be used to invoke FRK-V.

When calling the function `SRE`, we supplied the formula `f` containing information on the dependent variable and the covariates; the data (as a list that can include additional datasets); the BAUs; the basis functions; a flag `est_error`; and another flag `average_in_BAU`. The flag `est_error = TRUE` is used to estimate the measurement-error variance σ_ϵ^2 (where $\boldsymbol{\Sigma}_\epsilon \equiv \sigma_\epsilon^2 \mathbf{I}$) using semivariogram methods (Kang *et al.* 2009). At the time of writing, `est_error = TRUE` was only available for spatial data, not for spatio-temporal data. When not set to `TRUE`, each dataset needs to also contain a field `std`, the standard deviation of the measurement error (that can vary with the measurement).

FRK is built on the concept of a BAU, and hence the smallest spatial support of an observation has to be equal to that of a BAU. However, in practice, several datasets (such as the `meuse` dataset) are point-referenced. We reconcile this difference by assigning a support to every point-referenced datum equal to that of a BAU. Multiple point-referenced data falling within the same BAU are thus assumed to be noisy observations of the same random variable; see (6). As a consequence of this, when multiple observations fall into the same BAU, the matrices $\mathbf{V}_{\xi,Z}$ and $\mathbf{V}_{\delta,Z}$ will be sparse but not diagonal (since \mathbf{C}_Z will contain more than one non-zero element per column). This can increase the computational time required for estimation considerably. For large point-referenced datasets, such as the dataset considered in Section 4.2, it is reasonable to summarize the data at the BAU level. Since **FRK** is designed for use with large datasets, the argument `average_in_BAU` of the function `SRE` is defaulted to `TRUE`. In this default setting, all data falling into one BAU is averaged, and the standard deviation of the measurement error of the averaged data point is taken to be the average standard deviation of the measurement error of the individual data points. Consequently, the dataset is thinned. With large datasets and small BAUs, this thinning frequently does not cause performance degradation (see Section 4.2). Since the `meuse` dataset is relatively small, we set `average_in_BAU = FALSE`.

Step 5: The SRE model is fitted using the function `SRE.fit`. Maximum likelihood estimation is carried out using the EM algorithm of Section 2.2, which is assumed to have converged either when `n_EM` is exceeded or when the log-likelihood across subsequent steps does not change by more than `tol`. In this example, the EM algorithm converged in about 30 iterations; see Figure 5.

```
R> S <- SRE.fit(SRE_model = S, n_EM = 400, tol = 0.01, print_lik = TRUE)
```

Step 6: Finally, we predict at all the BAUs with the fitted spatial model. This is done using the function `predict`. The argument `obs_fs` dictates whether we attribute the fine-

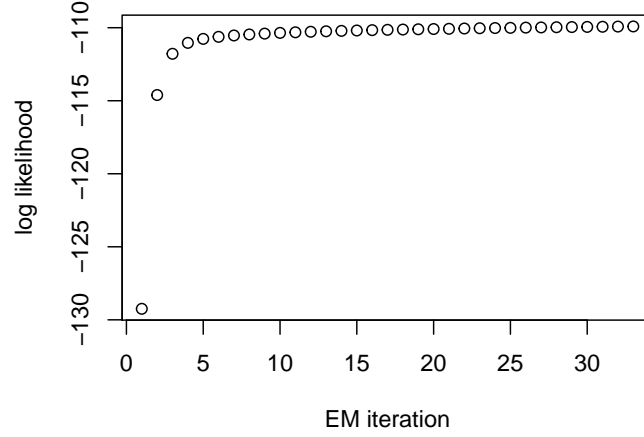


Figure 5: Convergence of the EM algorithm when using **FRK** with the `meuse` dataset.

scale variation to intra-BAU systematic error (Case 1) or to the process model (Case 2). In the code below, we use the default setting and allocate it to the process model.

```
R> GridBAUs1 <- predict(S, obs_fs = FALSE)
```

The object `GridBAUs1` now contains the prediction vector, the prediction standard error, and the square of the prediction standard error at the BAU level in the fields `mu`, `sd`, and `var`, respectively. These can then be visualized using standard plotting commands.

Predicting over larger polygons/areas

Now, assume that we wish to predict over regions encompassing several BAUs such that the matrix \mathbf{C}_P in (11) contains multiple non-zeros per row. We can create this larger regionalization by using the function `auto_BAUs` and specifying the cell size. This gives a ‘super-grid’ shown in Figure 6.

```
R> Pred_regions <- auto_BAUs(manifold = plane(), cellsize = c(600, 600),
+   type = "grid", data = meuse, convex = -0.05)
```

We carry out prediction on the super-grid by setting the `newdata` argument in the function `predict`, as given below.

```
R> Pred <- predict(S, newdata = Pred_regions, obs_fs = FALSE)
```

The predictions and the corresponding prediction standard errors on this super-grid are shown in Figure 6.

3.4. Computational considerations

While **FRK** beats the curse of ‘data dimensionality’ by dealing with matrices of size $r \times r$ instead of matrices of size $m \times m$, one must ensure that the number of basis functions, r , remains reasonably small. The reasons are two-fold. First, the computational time required to invert an $r \times r$ matrix increases cubically with r , and several such inversions are required

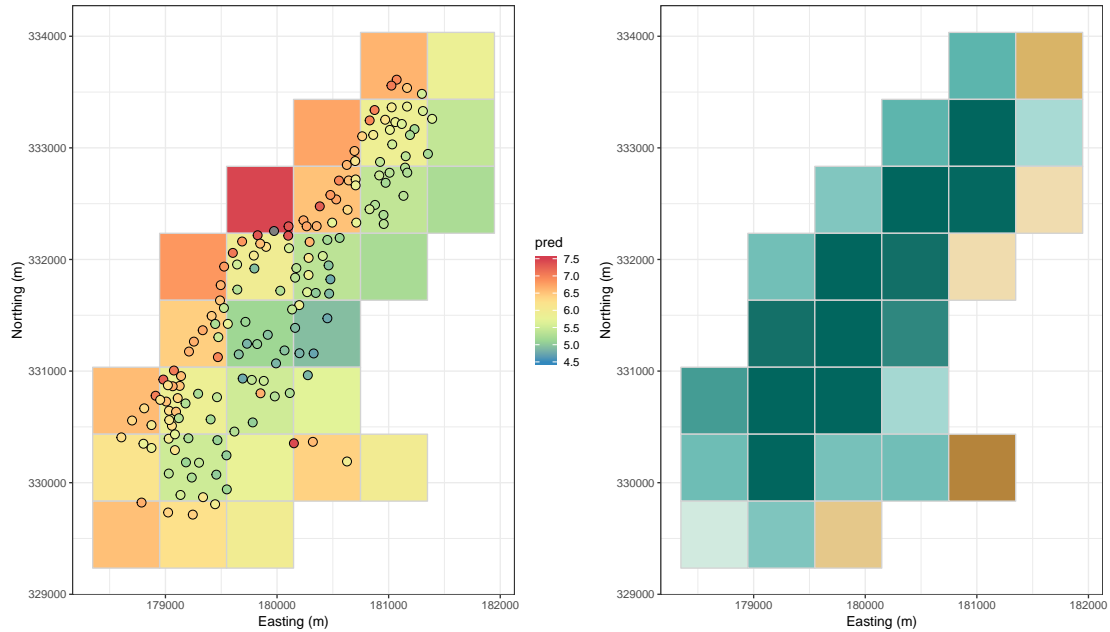


Figure 6: (Left panel) Prediction on a super-grid and overlaid observations for the **meuse** dataset. (Right panel) Prediction standard errors obtained with **FRK** from the **meuse** dataset over the super-grid. Both quantities are in logs of ppm.

when running the EM algorithm. Second, it is likely that more EM-algorithm iterations are required when r is large. In practice, r should not exceed a few thousand. The number of basis functions r can usually be controlled through the argument **nres**. The function **auto_basis** also takes an argument **max_basis** that automatically finds the number of resolutions required to not exceed the desired maximum number of basis functions.

The fitting and prediction algorithms scale linearly with the number of data points m and the number of BAUs N . However, if one has millions of data points, then the number of BAUs must exceed this and a big-memory machine will probably be required. Irrespective of problem size, we have noted considerable improvements in speed when using the **OpenBLAS** libraries (Wang, Zhang, Zhang, and Yi 2013).

Some of the operations in **FRK** can be run in parallel. To use a parallel back-end, one needs to set an option as follows:

```
R> opts_FRK$set("parallel", numcores)
```

where **numcores** is of class ‘integer’ (e.g., **numcores** = 4L to use 4 cores). When this option is set, the **parallel** package is used to set up a parallel backend using **makeCluster**, which is subsequently used for parallel operations. Currently, parallelism is limited in **FRK** to

- computing the integrals in (3) using Monte Carlo integration or, when appropriate, the approximation (4);
- finding which data are influenced by which BAUs and computing the weights in (6).

Unfortunately the EM algorithm, which is the bottleneck in a spatial analysis using FRK, is serial in nature and difficult to parallelize. Hence, `SRE.fit` takes as argument `method`, in recognition that in the future other, possibly parallelizable, estimation methods might be implemented to speed up the fitting process.

4. Comparison studies

In this section we compare the utility of the SRE model in **FRK** to standard kriging using **gstat**, and to two other popular models for modeling and predicting with large datasets in R: the LatticeKrig model that can be implemented with the package **LatticeKrig** (Nychka *et al.* 2019), and the SPDE-GMRF model that can be implemented with the package **INLA** (Lindgren and Rue 2015). In both these models the spatial field is decomposed as

$$\tilde{\lambda}(\mathbf{s}) = \sum_{l=1}^r \phi_l(\mathbf{s}) \eta_l; \quad \mathbf{s} \in D,$$

and \mathbf{K}^{-1} is modeled in such a way that it is sparse. These two models allow for feasible computation with large r , however neither includes an extra fine-scale effect $\xi(\cdot)$. The SPDE-GMRF model has the added interpretable feature that, for a given set of basis functions, \mathbf{K}^{-1} is such that the resulting field approximates a Gaussian process with a stationary covariance function from the Matérn class.

In Section 4.1, we first analyze a 2D simulated dataset. We shall see that while **FRK** may sometimes perform less well in terms of prediction accuracy due to the practical limit on the number of basis functions it uses, it does not under-fit (i.e., it gives valid results) since fine-scale variation is taken into account. In fact, we see that the SRE model in **FRK** provides better coverage in terms of prediction intervals, even with large datasets, when compared to other models that use considerably more basis functions but that do not account for fine-scale variation. In the second case study (Section 4.2), we consider three days of column-averaged carbon dioxide data from the Atmospheric InfraRed Sounder instrument on board the Aqua satellite .

4.1. A 2D simulated dataset

Let $D = [0, 1] \times [0, 1] \subset \mathbb{R}^2$, and consider a process $Y(\cdot)$ with covariance function $\text{COV}(Y(\mathbf{s}), Y(\mathbf{s} + \mathbf{h})) \equiv \sigma^2 \exp(-\|\mathbf{h}\|/\tau)$, where σ^2 is the marginal variance of the process and τ is the e-folding length-scale. Further, let m be the number of observations, and define the signal-to-noise ratio (SNR) to be the ratio of the marginal variance σ^2 to that of the measurement-error process, σ_ϵ^2 . In the inter-comparison, we consider cases where m is either 1,000 or 50,000, SNR is 0.2, 1, or 5, and τ is either 0.015 or 0.15. These choices of parameters help highlight the strengths and weaknesses of FRK with respect to the other approaches. For example, due to the relatively small number r of basis functions employed, we expect FRK to have lower prediction precision when the SNR is high and τ is low, but we expect the prediction intervals to be valid. We further split the domain into two side-by-side partitions, and we placed 95% of the observations in the left half (LH) and 5% in the right half (RH). This partitioning helps identify the different methods' capability of borrowing strength from a region with dense measurements to a region with sparse measurements. The measurement locations for the $m = 1,000$ case are shown in Figure 7, left panel.

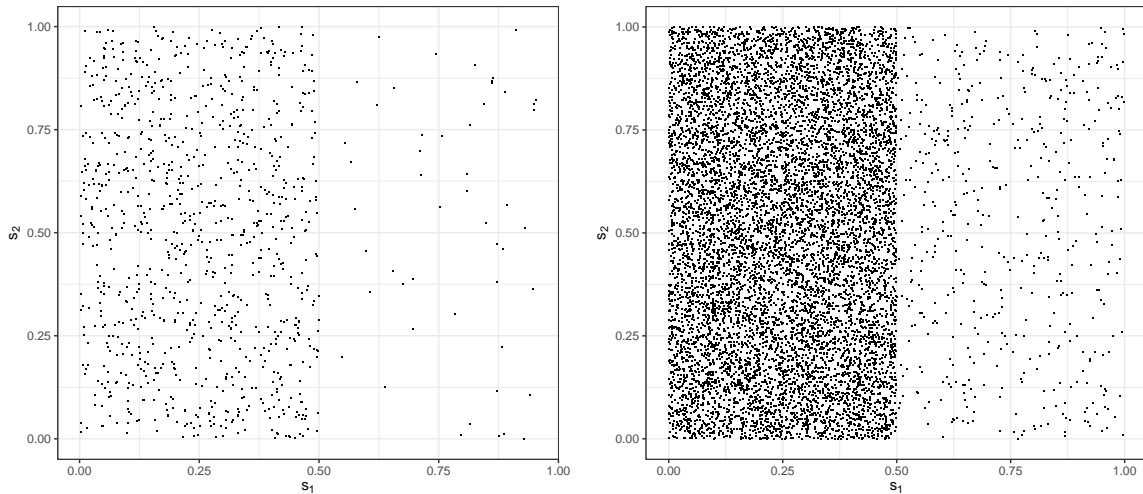


Figure 7: Measurement locations used in the experimental study. (Left panel) The number of locations $m = 1,000$. (Right panel) The number of locations $m = 10,000$. These locations are fixed, for a given m , across simulations in the experiment.

We simulated the process on a $1,000 \times 1,000$ grid using the package **RandomFields** (Schlather, Malinowski, Menck, Oesting, and Strokorb 2015). We used the 10^6 cells of the grid as our set of BAUs, D^G , and therefore each BAU was of size 0.001×0.001 . One such spatial-process realization for $\tau = 0.15$ and $\sigma^2 = 1$ is shown in Figure 8, left panel, while one with $\tau = 0.015$ and $\sigma^2 = 1$ is shown in Figure 8, right panel. With **gstat**, which we used to implement simple kriging (denoted *gstat*), we assumed the true underlying covariance function was known. Hence, when available (for the $m = 1,000$ case), the results of **gstat** should be taken as the gold standard. As m gets larger, simple kriging quickly becomes infeasible, since it is $O(m^3)$ in computational complexity.

We implemented the LatticeKrig model (denoted *LTK*) using the package **LatticeKrig**. We used `nlevel = 3` resolutions of Wendland basis functions, set the smoothness parameter `nu = 0.5`, and the number of grid points per spatial dimension at the coarsest resolution to `NC = 33`. The first resolution contained 1,849 basis functions, the second resolution contained 5,625, and the third resolution contained 19,321. In the case where $m = 1,000$, we set `findAwght = TRUE` for the effective process range to be estimated by maximum likelihood methods. Setting `findAwght = TRUE` was prohibitive for $m = 50,000$, but separate experiments showed that predictions from *LTK* were largely insensitive to this option for this value of m .

We implemented and fit the SPDE-GMRF model (denoted *SPDE*) using the package **INLA**. We constructed a triangular mesh using `inla.mesh.2d` with `max.edge = c(0.05, 0.05)` and `cutoff = 0.012`. This gave a mesh with a higher density of basis function on the left-hand side of the domain and (as with the LatticeKrig model) a buffer to reduce edge effects. The basis functions are defined by the triangles, and their number was around 2,500 for $m = 1,000$, and 5,000 for $m = 50,000$, while the parameter $\alpha = 3/2$ was used to reproduce Gaussian fields with a Matérn covariance function with smoothness parameter of $1/2$ (i.e., an exponential covariance function). Unlike **LatticeKrig** and **FRK**, **INLA** uses an approximate Bayesian method for inference, and thus it requires the specification of prior distributions of the parameters, for which we use penalized complexity priors (Fuglstad, Simpson, Lindgren,

and Rue 2019). For these simulation settings and our choice of prior distributions, we do not expect the inferential method to be a factor that largely influences the prediction and prediction errors.

For the SRE model implemented in **FRK** (denoted *FRK*) we put a block-exponential covariance structure on $\mathbf{K}_o(\boldsymbol{\theta})$ (`K_type` = "block-exponential"), and we set `nres` = 3, yielding, in total, 819 basis functions regularly distributed in the domain D . In this study we used **LatticeKrig** v7.0, **INLA** v18.07.12, and **FRK** v0.2.2.

For each configuration in the simulation experiment (i.e., the factorial design defined by $m \in \{1, 000, 50, 000\}$, $\text{SNR} \in \{0.2, 1, 5\}$, and $\tau \in \{0.015, 0.15\}$), we simulated $L = 100$ datasets. For prediction locations we took 1,000 locations at random on the left-hand side of the gridded domain D^G that coincided with measurement locations, and another 1,000 that did not; and we did the same for the right-hand side. When there were less than 1,000 measurement locations on a given side, all measurement locations were chosen as prediction locations for that side. The sets of locations are denoted as D_{LH}^O , D_{LH}^M , D_{RH}^O , and D_{RH}^M , respectively. These locations were kept constant across all simulation experiments for a given m .

In addition to the stationary, exponential, Gaussian process, we also simulated from the nonstationary process $Y^{\text{NS}}(\cdot)$, where

$$Y^{\text{NS}}(\mathbf{s}) = \frac{1}{2} \{Y_1(\mathbf{s}) \sin(2\pi s_1) \cos(2\pi s_2) + Y_2(\mathbf{s}) \sin(2\pi s_1)\}; \quad \mathbf{s} = (s_1, s_2)^\top \in D, \quad (14)$$

with $\text{COV}(Y_1(\mathbf{s}), Y_1(\mathbf{s}+\mathbf{h})) = \sigma_1^2 \exp(-\|\mathbf{h}\|/\tau_1)$ and $\text{COV}(Y_2(\mathbf{s}), Y_2(\mathbf{s}+\mathbf{h})) = \sigma_2^2 \exp(-\|\mathbf{h}\|^2/\tau_2)$. For this additional experiment, we set $m = 10, 000$, $\sigma_1 = \sigma_2 = 0.5$, and $\tau_1 = \tau_2 = 0.15$, and we used all configurations in the original experiment as described above. The measurement locations for this case are shown in Figure 7, right panel.

As prediction-performance measures ('responses' of the experiment), we considered the following:

- Root mean-squared prediction error: Let $\hat{Y}_X(\mathbf{s}; l)$ denote the 'model- X ' predictor of $Y(\mathbf{s}; l)$, where $Y(\mathbf{s}; l)$ is the l th simulated process evaluated at location \mathbf{s} and $X = \text{gstat}, \text{LTK}, \text{SPDE}, \text{FRK}$. Then the model- X predictor root-mean-squared prediction error (RMSPE) for the l th simulation is

$$\text{RMSPE}_X(l) \equiv \sqrt{\frac{1}{|D^*|} \sum_{\mathbf{s} \in D^*} \left(\hat{Y}_X(\mathbf{s}; l) - Y(\mathbf{s}; l) \right)^2}; \quad l = 1, \dots, L, \quad (15)$$

where $D^* = D_{\text{LH}}^O, D_{\text{LH}}^M, D_{\text{RH}}^O$, or D_{RH}^M . Since we are interested in benchmarking the model we use in **FRK**, we considered a measure of relative skill (RS), relative to *FRK*:

$$\text{RS}_X(l) \equiv \text{RMSPE}_X(l) / \text{RMSPE}_{\text{FRK}}(l); \quad l = 1, \dots, L,$$

where $X = \text{gstat}, \text{LTK}, \text{SPDE}$. Hence, $\text{RS} > 1$ (< 1) indicates that *FRK* has better (worse) prediction accuracy.

- Ninety-percent coverage: Let $\hat{\sigma}_X^2(\mathbf{s}; l)$ denote the prediction variance under model X . Then, the ninety-percent coverage, $I_X^{90}(l)$, denotes the percentage of times $Y(\mathbf{s}; l)$ lies in

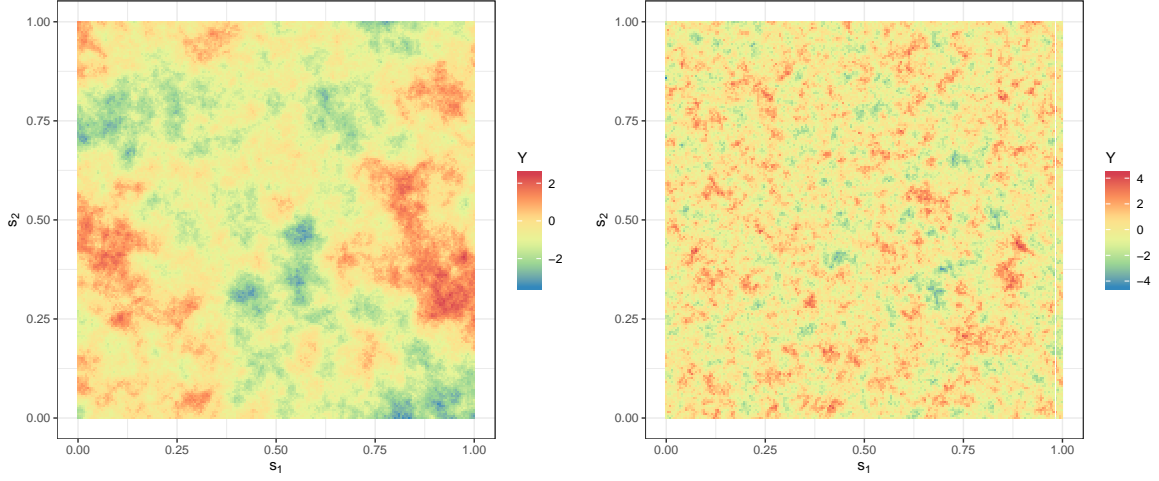


Figure 8: Simulations on a $1,000 \times 1,000$ grid D^G from a stationary Gaussian process with exponential covariance function $\text{COV}(Y(\mathbf{s}), Y(\mathbf{s} + \mathbf{h})) = \sigma^2 \exp(-\|\mathbf{h}\|/\tau)$. (Left panel) The e-folding length scale $\tau = 0.15$. (Right panel) The e-folding length scale $\tau = 0.015$.

the interval $[\hat{Y}_X(\mathbf{s}; l) - c\hat{\sigma}_X(\mathbf{s}; l), \hat{Y}_X(\mathbf{s}; l) + c\hat{\sigma}_X(\mathbf{s}; l)]$, over $\mathbf{s} \in D^*$. For the 90% coverage measure, c is the standard normal's inverse cumulative distribution function evaluated at $1 - (\frac{1}{2})(0.1) = 0.95$, which is 1.64 (rounded to two decimal places). That is,

$$I_X^{90}(l) \equiv \frac{1}{|D^*|} \sum_{\mathbf{s} \in D^*} \mathbb{I}(Y(\mathbf{s}; l) \in [\hat{Y}_X(\mathbf{s}; l) - c\hat{\sigma}_X(\mathbf{s}; l), \hat{Y}_X(\mathbf{s}; l) + c\hat{\sigma}_X(\mathbf{s}; l)]); \quad l = 1, \dots, L, \quad (16)$$

where $D^* = D_{\text{LH}}^O, D_{\text{LH}}^M, D_{\text{RH}}^O, D_{\text{RH}}^M$, and where $X = \text{gstat}, \text{LTK}, \text{SPDE}, \text{FRK}$.

We intentionally focus on coverage in order to highlight the strengths and weaknesses of the models in terms of uncertainty quantification. Other related measures, such as the interval score (Gneiting and Raftery 2007), penalize for both prediction interval width and poor coverage and are thus less suited to assess the issue of validity (i.e., whether the prediction intervals are correct, on average). The measures RS_X and I_X^{90} were considered for $\{Y(\mathbf{s}) : \mathbf{s} \in D^G\}$ simulated from the stationary process with exponential covariance function and from the nonstationary process $Y^{\text{NS}}(\cdot)$ in (14).

Distributions of RS_X for the original experiment with $m = 1,000$ and $m = 50,000$ are shown in Figures 9 and 10, respectively. While it is possible to proceed with an analysis of variance to analyze these results (e.g., Zhuang and Cressie 2014), here we discuss their most prominent features. First, when there are few ($m = 1,000$) data points (Figure 9), there is little difference between the methods for low SNR but, for high SNR, *SPDE* and *LTK* perform better in terms of RS when τ is small ($\tau = 0.015$; see, for example, the bottom left two panels of Figure 9). This was expected since the number of basis functions used begins to play an important role as the SNR increases (Zammit-Mangion, Cressie, and Shumack 2018). As expected, all prediction methods perform worse than or as well as, simple kriging with **gstat** (under a known covariance function).

The comparison in terms of RS is less clear when $m = 50,000$ (Figure 10). First, at unobserved locations, *FRK* is frequently outperformed in terms of RS by the other methods, since the

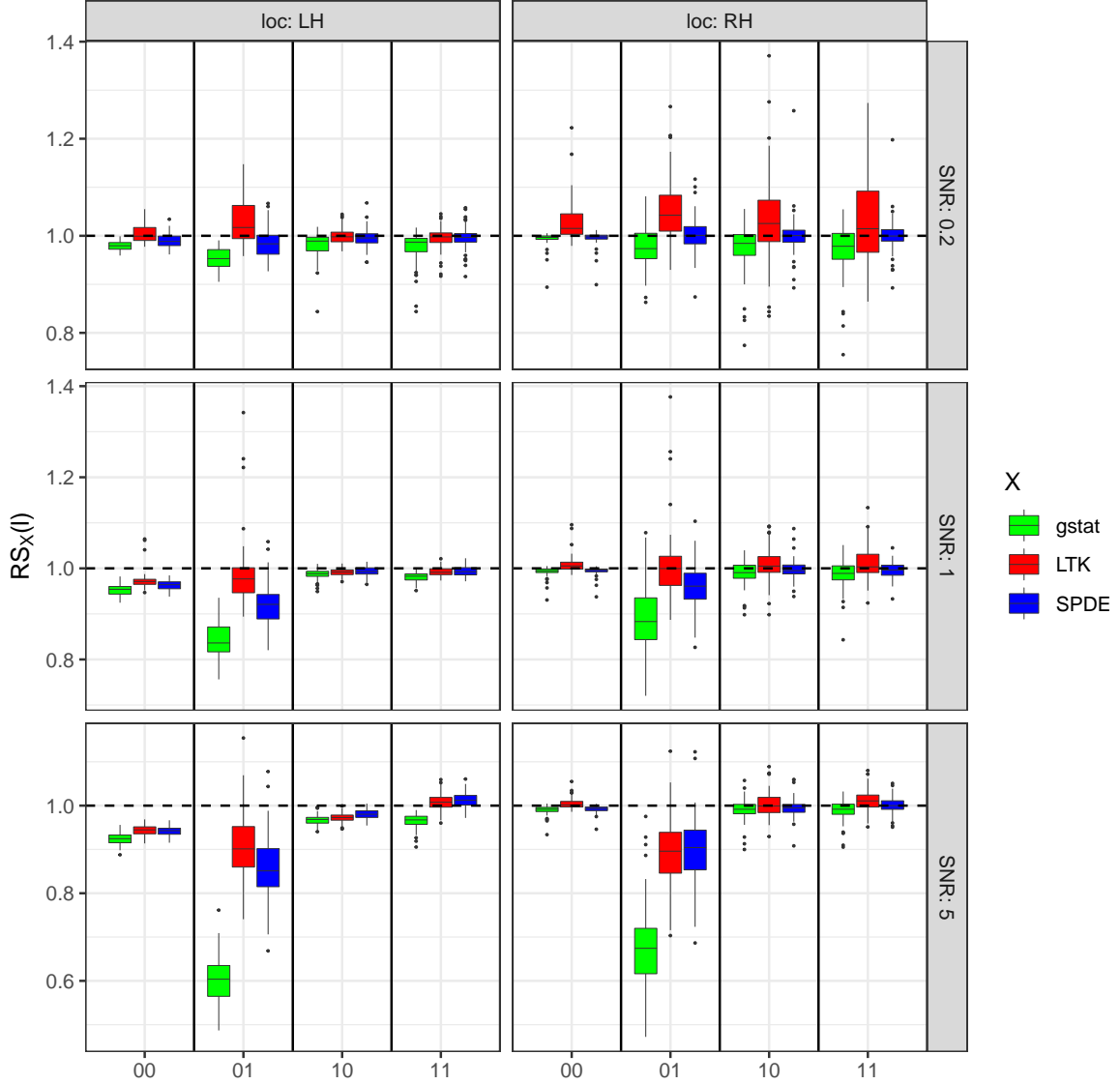


Figure 9: Boxplots of $\{RS_X(l) : l = 1, \dots, L\}$, $L = 100$, by model X (box color) for different SNR (signal-to-noise ratio), location (LH or RH), measurement-location coincidence, and process length scale, for the case where the number of data points $m = 1,000$. The x -axis labels are in the form ab where a is 0 if $\tau = 0.015$ and 1 if $\tau = 0.15$, and b is 0 if the prediction locations do not coincide with the measurement locations and 1 if they do. The boxes denote the interquartile range, the whiskers extend to the last values that are within 1.5 times the interquartile range from the quartiles, and the dots show the values that lie beyond the end of the whiskers. Values of RS_X larger (smaller) than 1 indicate superior (inferior) performance of *FRK*; a dashed line is shown at $RS_X = 1$.

relatively small number of basis functions is unable to adequately reconstruct the optimal (simple-kriging) predictor. On the other hand, at observed locations, the performance is SNR dependent and data-density dependent. In much of the design space, *FRK* performs worse (in terms of RS) than the other predictors at the measurement locations, but it begins to

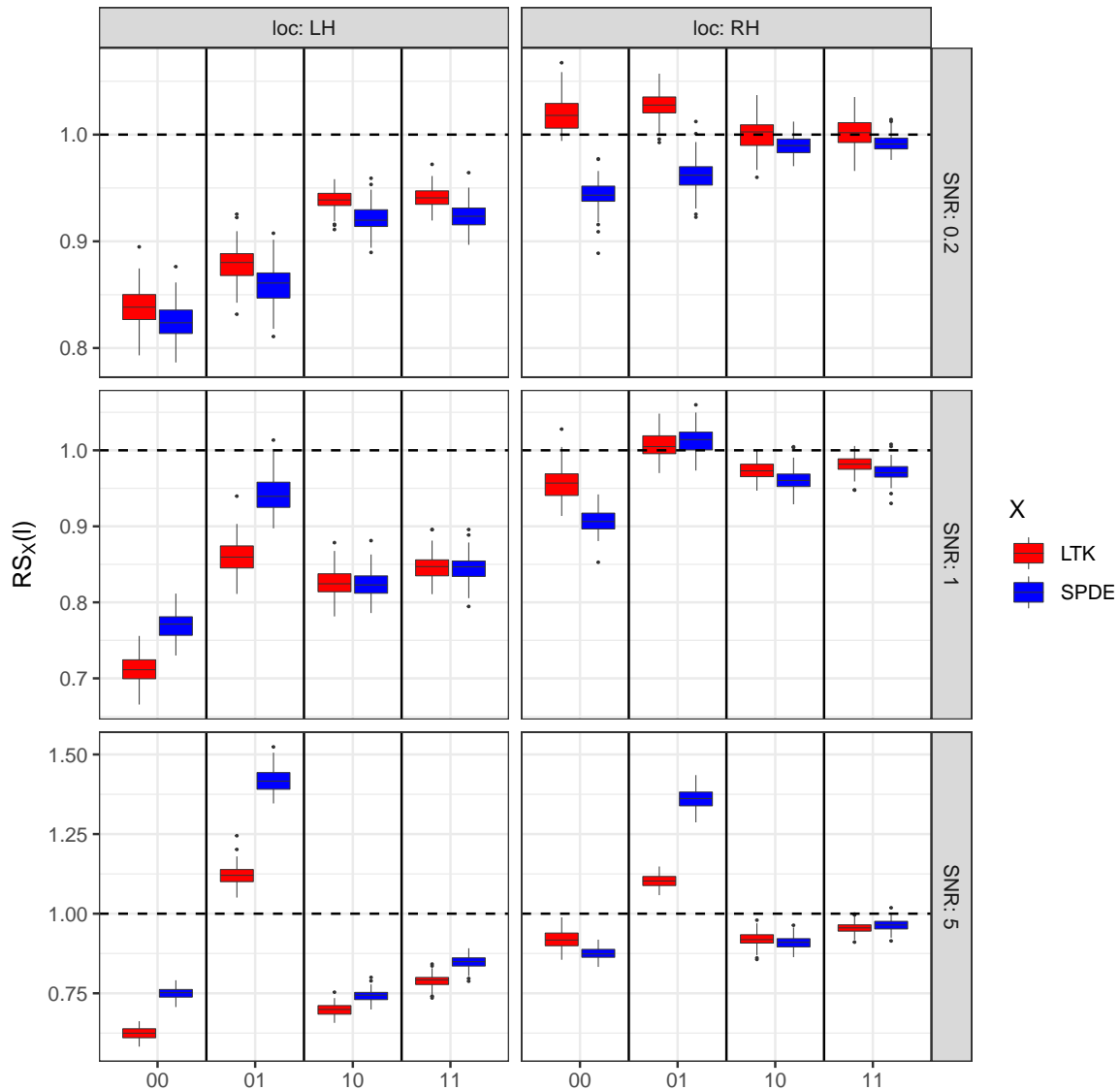


Figure 10: Same as Figure 9 but with the number of data points $m = 50,000$. Note that, in this case, simple kriging (with *gstat*) is computationally prohibitive and hence does not appear in the figure.

outperform *SPDE* and *LTK* as the SNR increases and when τ is small ($\tau = 0.015$). Now we turn to the question of ‘validity’ of the predictors.

An equally important performance measure to RMSPE is coverage. Distributions of I_X^{90} for $m = 1,000$ and $m = 50,000$ are shown in Figures 11 and 12, respectively. In the small-data case ($m = 1,000$), all methods are over-confident (more so in the left-hand part of the domain) and by varying degrees. In the large-data case ($m = 50,000$), both *SPDE* and *LTK* perform poorly in terms of coverage, providing over-confident predictions, especially when the SNR is large (SNR = 5). This is a result of these models relying on basis functions to reproduce the fine-scale variation and not attempting to separate out fine-scale variation from measurement error. The model implemented in **FRK** places a white-noise process at the BAU

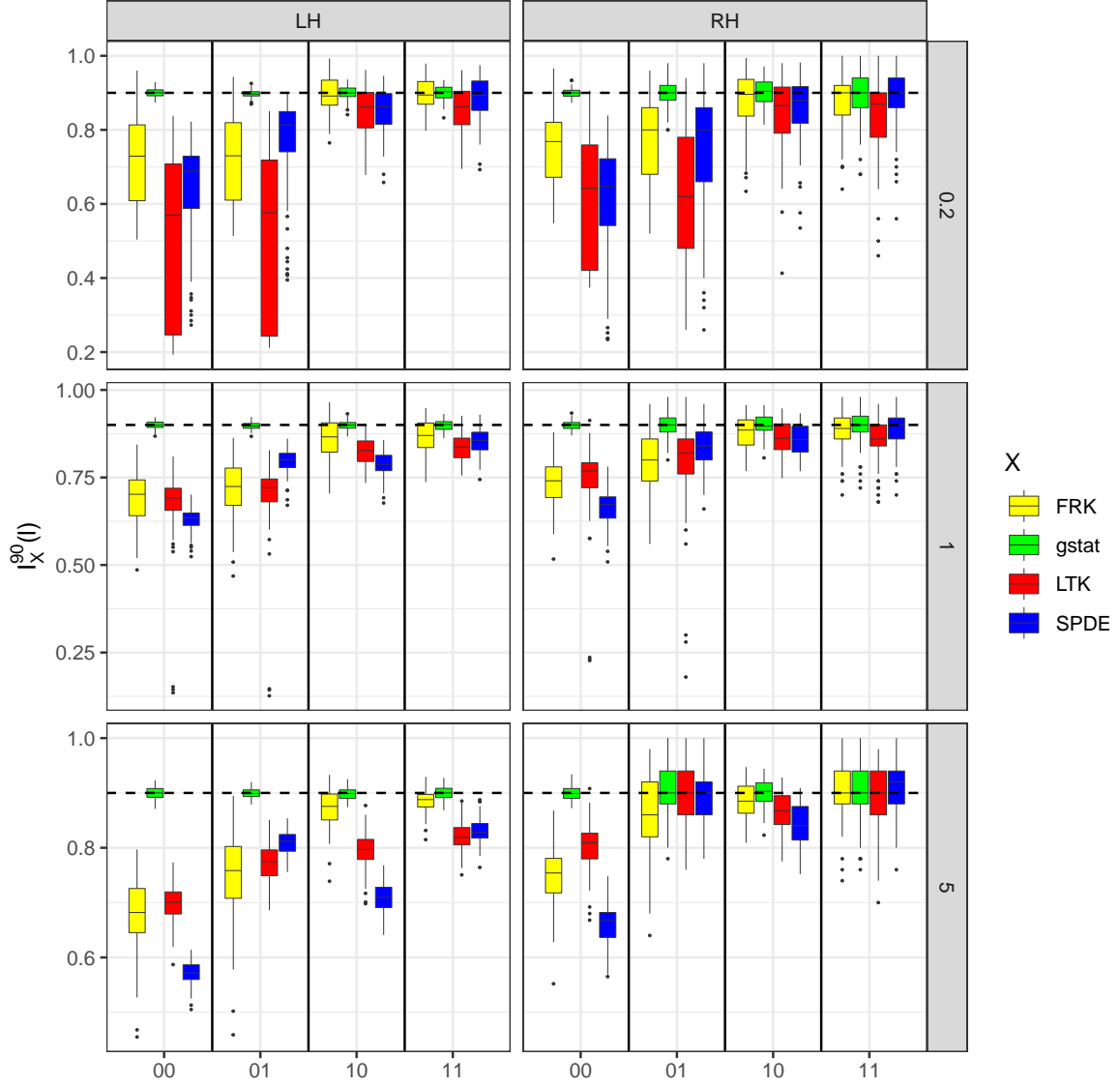


Figure 11: Boxplots of $\{I_X^{90}(l) : l = 1, \dots, L\}$, $L = 100$, by model X (box color) for different SNR (signal-to-noise ratio), location (LH or RH), measurement-location coincidence, and process length scale, for the case where the number of data points $m = 1,000$. The x -axis labels are in the form ab where a is 0 if $\tau = 0.015$ and 1 if $\tau = 0.15$, and b is 0 if the prediction locations do not coincide with the measurement locations and 1 if they do. The boxes denote the interquartile range, the whiskers extend to the last values that are within 1.5 times the interquartile range from the quartiles, and the dots show the values that lie beyond the end of the whiskers. The target value is 0.9 (dashed line). Values smaller than 0.9 indicate overconfidence of the prediction method.

level to capture the fine-scale variation and can thus yield good coverage despite the use of a relatively low-dimensional manifold. It is worth nothing that it is straightforward in **INLA** to include an extra fine-scale-variation term and fix the standard deviation of the measurement error to some pre-specified value, although this is rarely done. Here we are illustrating that

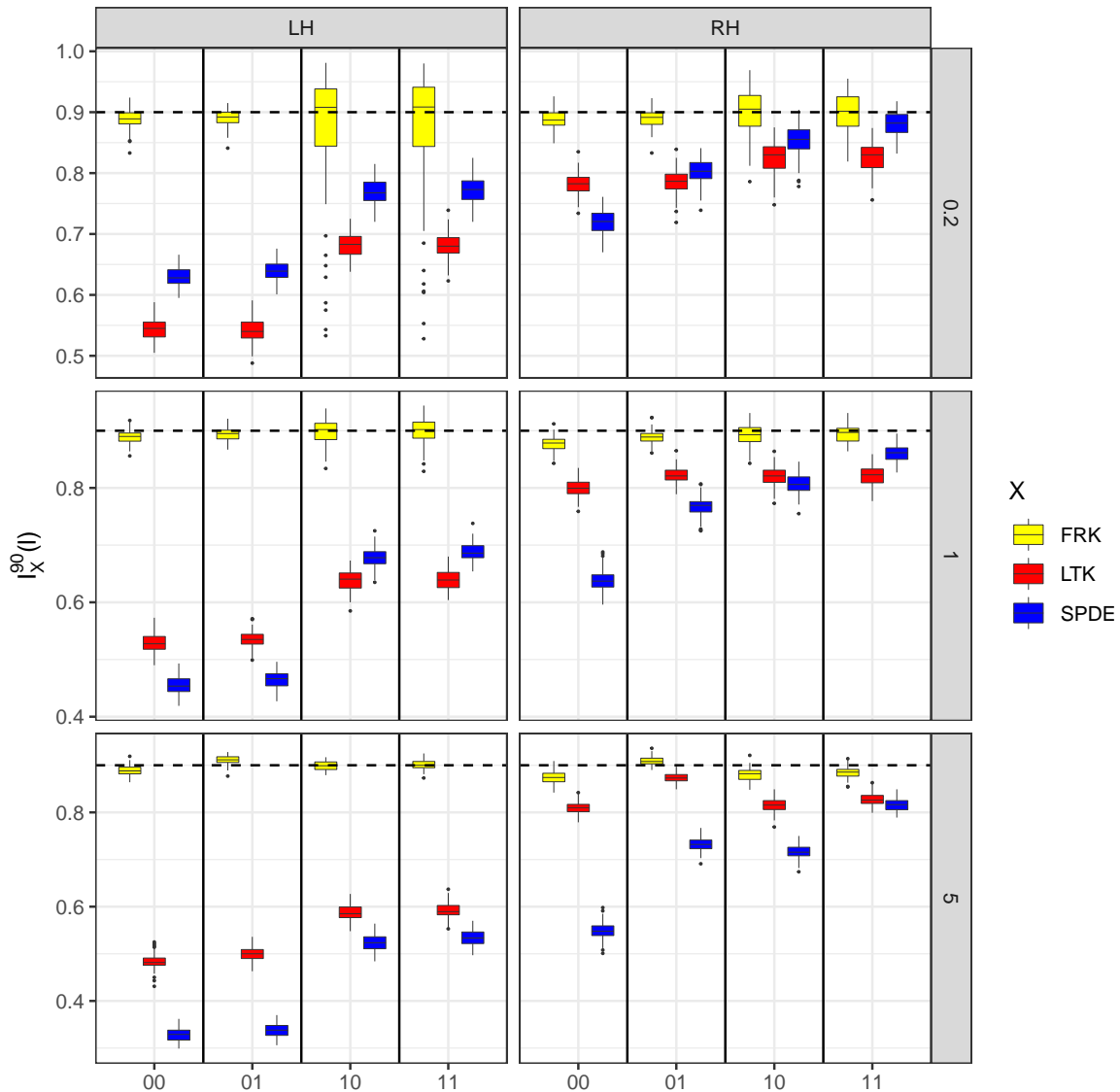


Figure 12: Same as Figure 11 but with the number of data points $m = 50,000$. Note that, in this case, simple kriging (with *gstat*) is computationally prohibitive and hence does not appear in the figure.

not doing this may lead to severe deleterious effects on coverage. The model used in **FRK** was also found to yield 90% interval scores that were at least as good as, or better than, the other two models for the case with $m = 50,000$ (results not shown).

To further investigate this issue, we re-ran the simulations and generated coverage diagnostics for predicted data, $\mathbf{Y}_P + \epsilon_P$, rather than for just \mathbf{Y}_P . The coverage for all methods was very good (results not shown), indicating that all methods are able to correctly apportion total variability. Consequently, these results show that inclusion of the fine-scale variation term is critical in reduced-rank approaches (irrespective of the number of basis functions) with large datasets when predicting the hidden process. (It is not critical when predicting missing data). The simple semivariogram method employed by **FRK** for estimating the measurement-error

	RMSPE (LH obs)	RMSPE (LH unobs)	RMSPE (RH obs)	RMSPE (RH unobs)	90% CR (LH obs)	90% CR (LH unobs)	90% CR (RH obs)	90% CR (RH unobs)
LTK	0.13	0.12	0.24	0.25	0.86	0.87	0.88	0.87
SPDE	0.13	0.12	0.24	0.25	0.90	0.89	0.91	0.88
FRK	0.13	0.13	0.24	0.25	0.92	0.92	0.91	0.91

Table 3: Root mean squared prediction error (RMSPE) and 90% coverage (CR) for the case where $m = 10,000$ data were simulated from the nonstationary process $Y^{\text{NS}}(\cdot)$.

variance is a step in the right direction, and it appears to yield good results in the first instance. However, ideally, the standard deviation of the measurement error is known from the application and fixed *a priori*.

Overall, all models have their own relative strengths and weaknesses, largely arising from the differences in (i) the type and number of basis functions employed, and (ii) the presence or otherwise of a fine-scale-process term. In this experiment we saw that the model employed by **FRK** produces predictions that are valid, on average. However for large-data situations, our experiment shows *FRK* predictions to be less efficient, as expected due to the restriction on the number of basis functions that can be used.

In the nonstationary case (14), all methods performed similarly, with *LTK* being slightly overconfident and *FRK* being slightly underconfident; see Table 3. This similarity is not surprising since in (14) we set $\tau_1 = \tau_2 = 0.15$, which results in a process that is highly spatially correlated as well as rather smooth. The resulting process has a similar overall length scale and SNR to that simulated in the original experiment that yielded the results shown in the second row (SNR = 1) of the third (LH, ‘10’), fourth (LH, ‘11’), seventh (RH, ‘10’), and eighth (RH, ‘11’) columns of Figures 9 and 11. We see that all three methods performed similarly, and satisfactorily, in this case.

4.2. Modeling and prediction with data from the AIRS instrument

The US National Aeronautics and Space Administration (NASA) launched the Aqua satellite on 2002-05-04, with several instruments on board, including the Atmospheric Infrared Sounder (AIRS). AIRS retrieves column-averaged CO₂ mole fraction (in units of parts per million), denoted XCO₂ (with particular sensitivity in the mid-troposphere), amongst other geophysical quantities (Chahine *et al.* 2006). The data we shall use consist of XCO₂ measurements taken between 2003-05-01 and 2003-05-03 (inclusive). These data are a subset of those available with **FRK**. We compare *LTK*, *SPDE*, and *FRK* on the three-day AIRS dataset, and we assess the utility of the methods on a validation dataset that we hold out.

Modeling on the sphere with **FRK** proceeds in a very similar fashion to the plane, except that a coordinate reference system (CRS) on the surface of the sphere needs to be declared for the data. This is implemented using a ‘CRS’ object with string “+proj=longlat +ellps=sphere”. We next outline the six steps required to fit these data using **FRK**.

Step 1: Fifteen days of XCO₂ data from AIRS (in May 2003) are loaded by using the command `data("AIRS_05_2003", package = "FRK")`. In this case study, we subset the data to include only the first three days, which contains 43,059 observations of XCO₂ in parts per million (ppm). We subsequently divide the data into a training dataset of 30,000 observations, chosen at random (`AIRS_05_2003_t`), and a validation dataset (`AIRS_05_2003_v`) containing

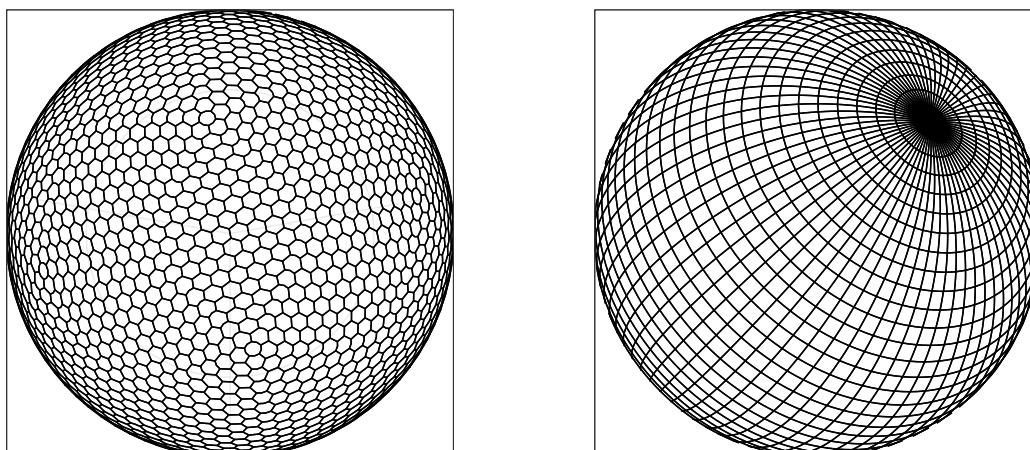


Figure 13: BAUs constructed on the surface of the sphere. (Left panel) BAUs are the ISEA3H hexagons (for visualization purposes we show resolution 5). (Right panel) BAUs are a $5^\circ \times 5^\circ$ lon-lat grid.

the remaining observations. To instruct **FRK** to fit the SRE model on the surface of a sphere, we assign the appropriate ‘CRS’ object to the data as follows:

```
R> coordinates(AIRS_05_2003) <- ~lon + lat
R> proj4string(AIRS_05_2003) <- CRS("+proj=longlat +ellps=sphere")
```

Step 2: The next step is to create BAUs. This is done using the `auto_BAUs` function but this time with the manifold specified to be the surface of a sphere with radius equal to that of Earth. We also specify that we wish the BAUs to form an icosahedral Snyder equal area aperture 3 hexagon (ISEA3H) discrete global grid (DGG) at resolution 9 (for a total of 186,978 BAUs). Resolutions 0–6 are included with **FRK**; higher resolutions are available in the package **dggrids** (Zammit-Mangion 2020). By default, this will create a hexagonal grid on the surface of the sphere, however it is also possible to have the more traditional lon-lat grid by specifying `type = "grid"` and declaring a `cellsize` in units of degrees. An example of an ISEA3H grid, at resolution 5 (which would yield a total of 6,910 BAUs), is shown in Figure 13, left panel, while a $5^\circ \times 5^\circ$ lon-lat grid using `type = "grid"` is shown in Figure 13, right panel.

```
R> isea3h_sp_poldf <- auto_BAUs(manifold = sphere(), isea3h_res = 9,
+   type = "hex", data = AIRS_05_2003)
R> isea3h_sp_poldf$fs <- 1
```

Step 3: Now the basis functions are constructed, again of type "bisquare", with three resolutions, to yield a total of 1,176 basis functions; see Figure 14, which was generated using the function `show_basis`.

```
R> G <- auto_basis(manifold = sphere(), data = AIRS_05_2003_t, nres = 3,
+   isea3h_lo = 2, type = "bisquare")
```

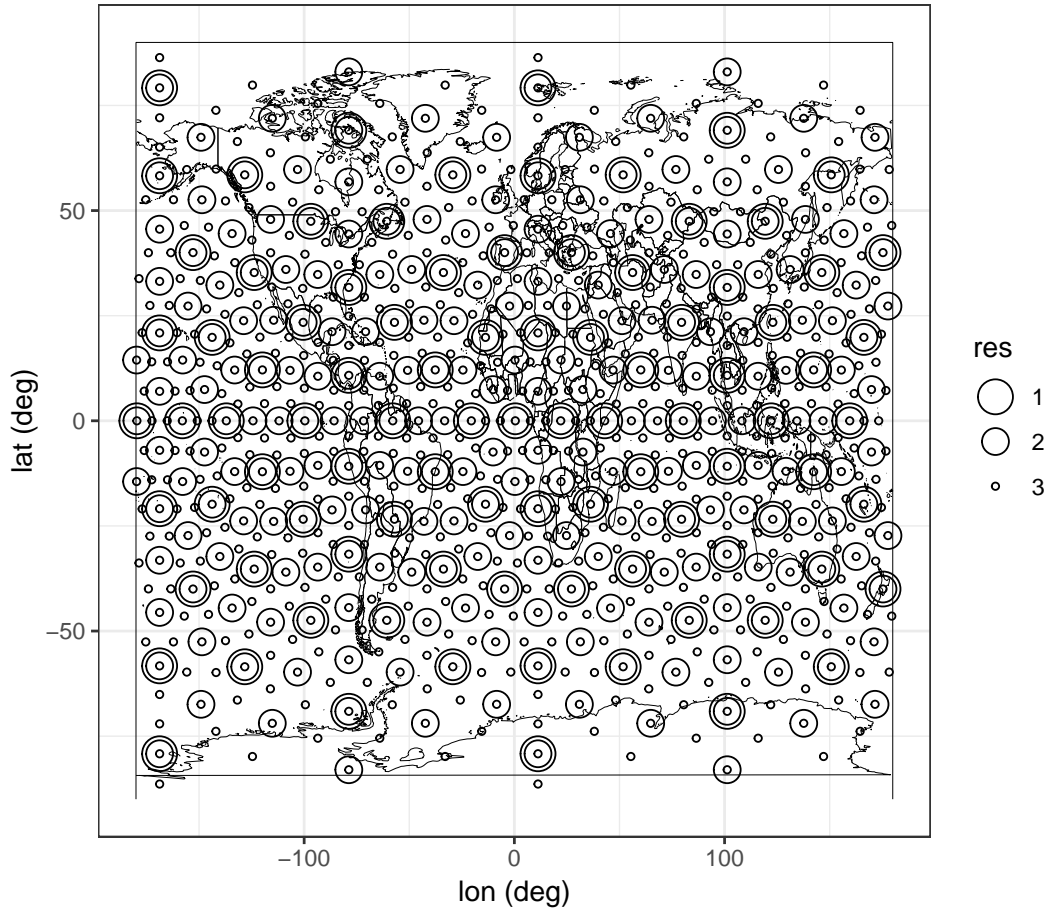


Figure 14: Basis functions used in modeling and predicting with the AIRS data. The basis-function centroids are constructed using the function `auto_basis`. The diameter of the circles are indicative of the resolution the basis functions belong to.

Steps 4–5: Since XCO_2 , a column-averaged CO_2 mole fraction in units of ppm, has a latitudinal gradient, we use latitude as a covariate in our model. The SRE object is then constructed in the same way as in Section 3.3. The AIRS footprint is approximately 50 km in diameter, which is smaller than the BAUs we use (approximately 100 km in diameter), and hence it is possible that multiple observations fall into the same BAU. Recall from Section 3.3 (Step 4) that when multiple data points fall into the same BAU that these are correlated through either intra-BAU systematic error (Case 1) or fine-scale process variation at the BAU level (Case 2). However, recall also that when multiple observations fall into the same BAU, the matrices $\mathbf{V}_{\xi, Z}$ and $\mathbf{V}_{\delta, Z}$ are sparse but not diagonal, and this can increase computational time considerably. For large datasets in which each datum has relatively small (relative to the BAU) spatial support, such as the AIRS dataset, it is frequently reasonable to let the argument `average_in_BAU = TRUE` (as it is by default) to indicate that one wishes to summarize the data at the BAU level.

In the code below we implement FRK using the default case (Case 2, where $\sigma_\delta^2 = 0$ and σ_ξ^2 is estimated).

	INLA	LTK	FRK-Ma	FRK-Mb	FRK-V
RMSPE	3.08	3.10	3.14	3.13	3.16
CRPS	1.71	1.72	1.74	1.73	1.75
90% coverage	0.91	0.91	0.90	0.90	0.89
Time (s)	74.00	336.00	1122.00	270.00	94.00

Table 4: Root-mean-squared prediction error (RMSPE), continuous-ranked probability score (CRPS), and computational times for the different methods (see text for details). Computational times for **INLA** report the time taken to compute the `inla()` function; for **LatticeKrig** the time to fit the model, predict and generate 30 conditional simulations; and for **FRK** the time to generate the basis functions, the BAUs, the EM estimates of the SRE models (with an EM convergence tolerance of 0.1) and predictions.

```
R> f <- co2avgret ~ lat
R> S <- SRE(f = f, list(AIRS_05_2003_t), basis = G, BAUs = isea3h_sp_poldf,
+   est_error = TRUE, average_in_BAU = FALSE)
R> S <- SRE.fit(SRE_model = S, n_EM = 1000, tol = 0.1, print_lik = FALSE)
```

Step 6: To predict at the BAU level, we invoke the `predict` function.

```
R> pred_isea3h <- predict(S)
```

The prediction and prediction standard error maps obtained using **FRK**, together with the observations, are shown in Figure 15. We denote the implementation above of FRK as FRK-Ma, where “M” denotes the case for the modeled $\text{VAR}(\boldsymbol{\eta}) = \mathbf{K}_o(\boldsymbol{\vartheta})$ and “a” denotes the case for `average_in_BAU` set to `FALSE`.

We evaluated the utility of the SRE model used in **FRK**, the LatticeKrig model (with **LatticeKrig**), and the SPDE-GMRF model (with **INLA**), using out-of-sample prediction at the validation-data locations. We also re-ran **FRK** with `average_in_BAU` set to `TRUE` (denoted FRK-Mb) and `K_type = "unstructured"` (FRK-V). With **INLA** we approximated an SPDE with $\alpha = 2$ on a global mesh of 6,550 basis functions and used penalized complexity prior distributions for the parameters (Fuglstad *et al.* 2019). With **LatticeKrig** we used three resolutions with a total of 12,703 basis functions on \mathbb{R}^2 using the lon-lat coordinates to denote spatial locations. As comparison measures we used the RMSPE (15) between the validation values and their respective predicted observations, the continuous ranked probability score (CRPS, Gneiting, Balabdaoui, and Raftery 2007), and the actual coverage of a 90% prediction interval (16) but obtained with respect to the validation data instead of the process Y at the validation-data locations.

The results are summarized in Table 4. In this example, we see that there is little practical difference in performance between the five methods despite the large difference in the number of basis functions and the form of the models; FRK performs about 2% worse than the others in terms of RMSPE. As expected, since we are validating against data (and not against the true process, which is unknown here), all methods perform acceptably in capturing total variation. However, the FRK methods gave prediction standard errors of the process that were, on average, double those provided by *LTK* and *SPDE*. This mirrors what was seen in Section 4, where *SPDE* and *LTK* were generally overconfident, although in this case the

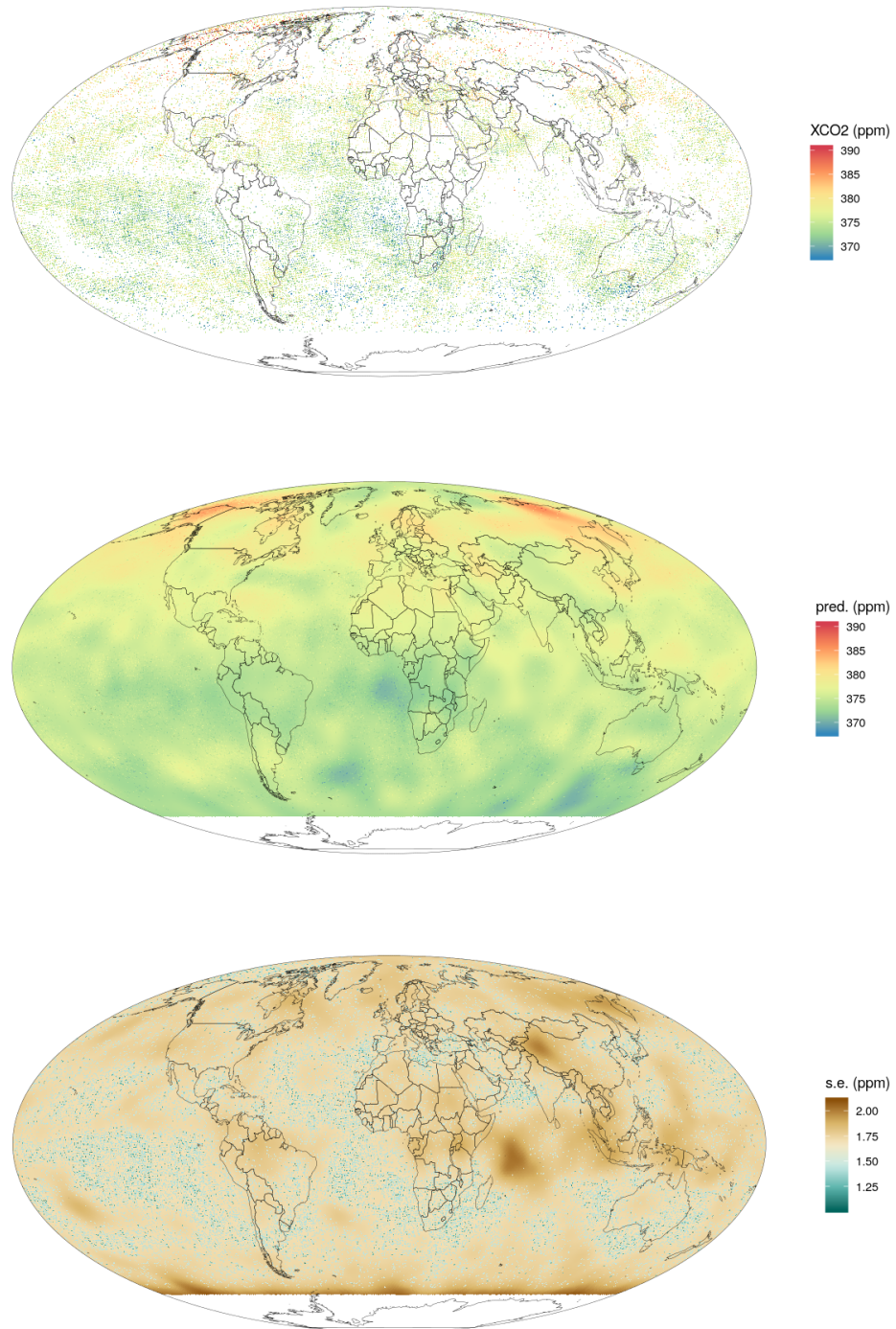


Figure 15: (Top panel) XCO₂ data in parts per million (ppm) from the AIRS instrument between 2003-05-01 and 2003-05-03 (inclusive). (Middle panel) Prediction of \mathbf{Y}_P in ppm using **FRK**. (Bottom panel) Prediction standard error of \mathbf{Y}_P in ppm using **FRK**. Note that AIRS does not release data at latitudes below 60°S.

true process is unknown and one can only speculate the reasons for the validation errors' behaviors. Nevertheless, the simulations in Section 4.1 indicate FRK's accurate coverage of the process Y .

Computation time for all three packages were similar under the chosen configurations (except for FRK-Ma that assumes intra-BAU correlations). For **FRK**, we computed the predictions and prediction standard errors directly using sparse-matrix operations, while we used **INLA**'s predictor functionality to obtain the prediction and prediction standard errors for the SPDE-GMRF model. We obtained **LatticeKrig**'s prediction errors using 30 conditional simulations.

5. Change-of-support, anisotropy, and custom basis functions

Sections 1–4 introduced the core spatial functionality of **FRK**. The purpose of this section is to present additional functionality that may be of use to the spatial analyst.

5.1. Multiple observations with different supports

In **FRK**, one can make use of multiple datasets with different spatial supports with little difficulty. Consider the `meuse` dataset. We synthesize observations with a large support by changing the `meuse` object into a `'SpatialPolygonsDataFrame'`, where each polygon is a square of size $300 \text{ m} \times 300 \text{ m}$ centered around the original `meuse` data point (see Figure 16, left panel). For reference, the constructed BAUs are of size $50 \text{ m} \times 50 \text{ m}$. Once this object is set up, which we name `meuse_pols`, we assign zinc values to the polygons by fitting a spherical semi-variogram model to the log zinc concentrations in the original `meuse` dataset, generating a realization by conditionally simulating once at the BAU centroids, exponentiating the simulated values, aggregating accordingly, and adding on measurement error with variance 0.01. The analysis proceeds in precisely the same way as in Section 3, but with `meuse_pols` used instead of `meuse`.

The predictions and the prediction standard errors using `meuse_pols` are shown in Figure 16, center and right panels, respectively. We note that the supports of the observations and the

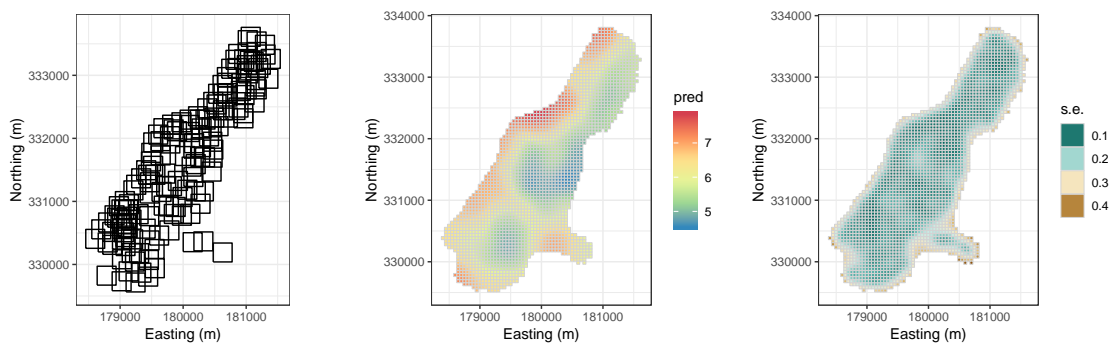


Figure 16: Data footprints, predictions and prediction standard errors obtained with **FRK** using the `meuse` dataset in logs of ppm, where each observation (black square) is assumed to have a spatial footprint of $300 \text{ m} \times 300 \text{ m}$. The BAUs are $50 \text{ m} \times 50 \text{ m}$ in size. (Left panel) The spatial footprints of the synthesized data. (Center panel) FRK predictions at the BAU level. (Right panel) FRK prediction standard errors at the BAU level.

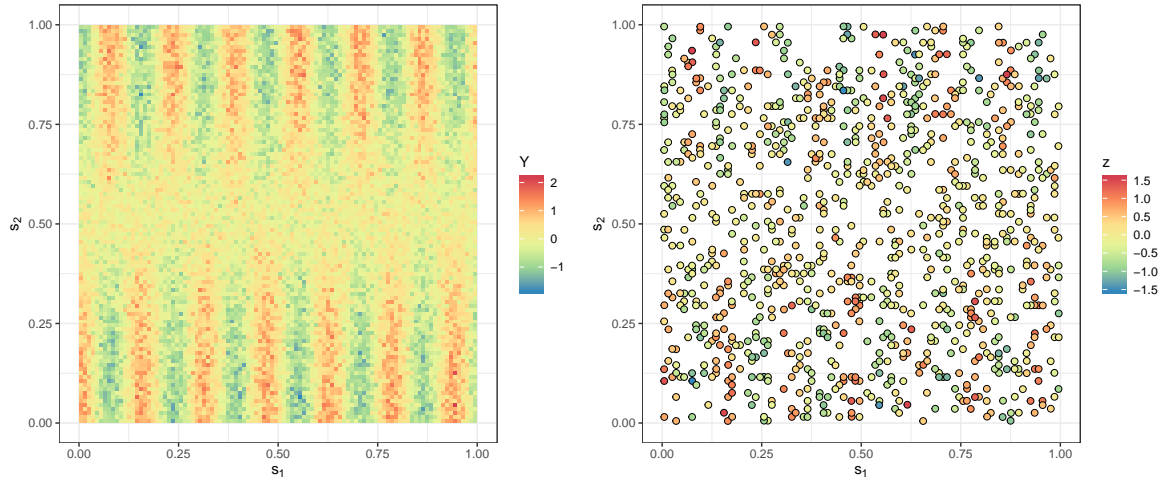


Figure 17: FRK with anisotropic fields. (Left panel) Simulated process. (Right panel) Observations at 1,000 randomly chosen grid points.

BAUs do not precisely overlap: Recall that, for simplicity, we assumed that an observation is taken to overlap a BAU if and only if the centroid of the BAU lies within the observation footprint. (A refinement of this simple method will require a more detailed consideration of the BAU and observation footprint geometry and is the subject of future work.)

5.2. Anisotropy: Changing the distance measure

Highly non-stationary and anisotropic fields may be easier to model on a deformed space on which the process is approximately stationary and isotropic (e.g., [Sampson and Guttorp 1992](#); [Kleiber 2016](#)). In **FRK**, a deformation can be introduced to capture geometric anisotropy by changing the distance measure associated with the manifold. As an illustration, we simulated an anisotropic, noisy, spatial process on a fine grid in $D = [0, 1] \times [0, 1]$ and assumed that 1,000 randomly-located grid points were observed. The process and the sampled data are shown in Figure 17.

In this simple case, to alter the modified distance measure we note that the spatial frequency in x is approximately four times that in y . Therefore, in order to generate anisotropy, we use a measure that scales x by 4. In **FRK**, a ‘measure’ object requires a distance function and the dimension of the manifold on which it is used, and it is constructed as follows:

```
R> dist_fun <- function(x1, x2 = x1) {
+   scaler <- diag(c(4, 1))
+   distR(x1 %*% scaler, x2 %*% scaler)
+ }
R> asymm_measure <- measure(dist = dist_fun, dim = 2L)
```

The distance function can be assigned to the manifold as follows.

```
R> TwoD_manifold <- plane(measure = asymm_measure)
```

We now generate a grid of basis functions (here at a single resolution) manually. First, we create a 5×14 grid on D , which we will use as centers for the basis functions. We then call

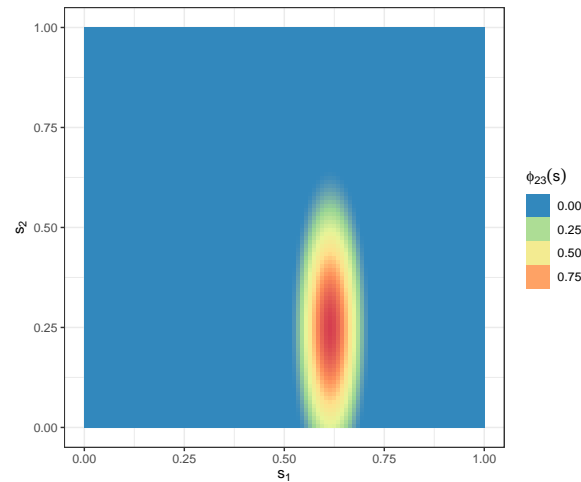


Figure 18: Basis-function number 23 of the 70 spatial basis functions constructed to fit an anisotropic spatial field.

the function `local_basis` to construct bisquare basis functions centered at these locations with a range parameter (i.e., the radius in the case of a bisquare) of 0.4. Due to the scaling used, this implies a range of 0.1 in x and a range of 0.4 in y . Basis-function number 23 is shown in Figure 18.

```
R> basis_locs <- expand.grid(seq(0, 1, length = 14), seq(0, 1, length = 5))
R> G <- local_basis(manifold = TwoD_manifold, loc = as.matrix(basis_locs),
+   scale = rep(0.4, nrow(basis_locs)), type = "bisquare")
```

From here on, the analysis proceeds in exactly the same way as given in the other examples. The predictions and prediction standard errors are shown in Figure 19. Note that complicated distance functions need to be obtained offline, for instance using multi-dimensional scaling (Sampson and Guttorp 1992).

5.3. Customized basis functions and BAUs

The package **FRK** provides the functions `auto_BAUs` and `auto_basis` to help the user construct the BAUs and basis functions based on the data that are supplied. However, these could be done manually. The object containing the basis functions needs to be of class ‘`Basis`’ that defines 5 slots:

- **dim**: The dimension of the manifold.
- **fn**: A list of functions. By default, distances used in these functions (if present) are attributed to a manifold, but arbitrary distances can be used.
- **pars**: A list of parameters associated with each basis function. For the local basis functions used in this paper (constructed using `auto_basis` or `local_basis`), each list item is another list with fields `loc` and `scale` where `length(loc)` is equal to the dimension of the manifold and `length(scale)` is equal to 1.

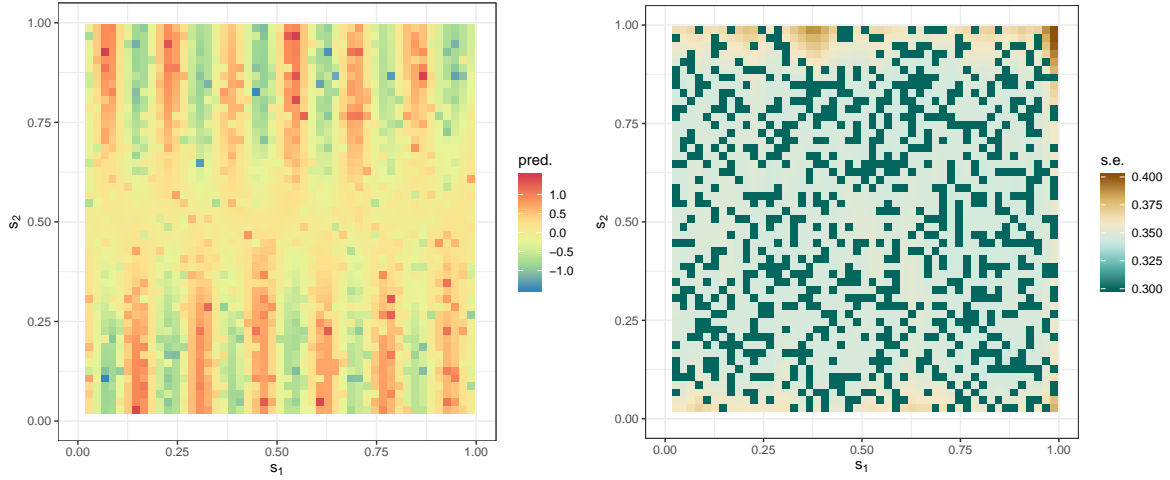


Figure 19: FRK using data shown in Figure 17 (generated by an anisotropic field). (Left panel) FRK predictions. (Right panel) FRK prediction standard errors.

- **df**: A data frame with number of rows equaling the number of basis functions and containing auxiliary information about the basis functions (e.g., resolution number).
- **n**: An integer equal to the number of basis functions.

The constructor **Basis** can be used to instantiate an object of this class.

There are less restrictions for constructing BAUs; for spatial applications, they are usually either ‘**SpatialPixelsDataFrame**’s or ‘**SpatialPolygonsDataFrame**’s. In a spatio-temporal setting, the BAUs need to be of class ‘**STFDF**’, where the spatial component is usually either a ‘**SpatialPixels**’ or a ‘**SpatialPolygons**’ object. In either case, the **data** slot of the object must contain

- all covariates used in the model;
- a field **fs** containing elements proportional to the fine-scale variation at the BAU level; and
- fields that can be used to summarize the BAU as a point, typically the centroid of each polygon. The names of these fields need to equal those of the **coordnames**(BAUs) (typically **c("x", "y")** or **c("lon", "lat")**).

6. Spatio-temporal FRK

Fixed rank kriging in space and time is different from fixed rank filtering (Cressie, Shi, and Kang 2010), where a temporal autoregressive structure is imposed on the temporally evolving basis-function weights $\{\eta_t\}$, and where Rauch-Tung-Striebel smoothing is used for inference on $\{\eta_t\}$. In **FRK**, the basis functions can also have a temporal dimension; then the only new aspect beyond the purely spatial analysis given above is specifying these spatio-temporal basis functions. We describe how this can be done in Section 6.1. In Section 6.2 we show

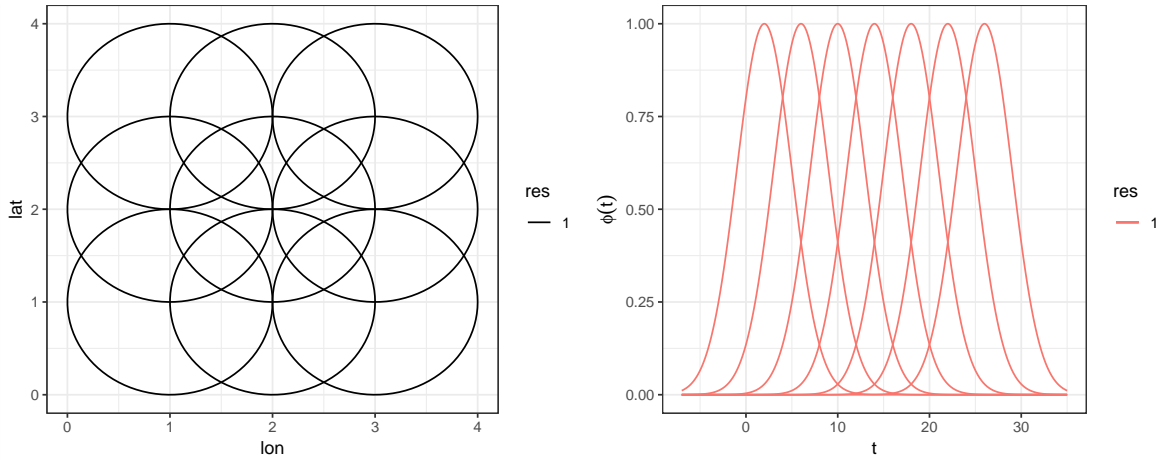


Figure 20: Basis functions used to construct the spatio-temporal basis functions. (Left panel) Locations of bisquare spatial basis functions (circles represent the apertures, here all equal). (Right panel) Temporal basis functions.

how this can be applied to modeling and prediction with data from the more recent Orbiting Carbon Observatory-2 (OCO-2) satellite that measures XCO_2 .

6.1. Basis-function construction

FRK allows for kriging in space and time through the use of spatio-temporal basis functions constructed through the tensor product of spatial basis functions with temporal basis functions. Specifically, consider a set of r_s spatial basis functions $\{\phi_p(\mathbf{s}) : p = 1, \dots, r_s\}$ and a set of r_t temporal basis functions $\{\psi_q(t) : q = 1, \dots, r_t\}$. Then we construct the set of spatio-temporal basis functions as $\{\phi_{st,pst}(\mathbf{s}, t) : pst = 1, \dots, r_s r_t\} = \{\phi_p(\mathbf{s})\psi_q(t) : p = 1, \dots, r_s; q = 1, \dots, r_t\}$.

To illustrate their construction, consider the following set of spatial basis functions.

```
R> centroids <- as.matrix(expand.grid(x = 1:3, y = 1:3))
R> G_spatial <- local_basis(manifold = plane(), loc = centroids,
+   scale = rep(2, 9), type = "bisquare")
```

The function call above returns a set of bisquare basis functions centered at `loc` with aperture equal to `scale`. The same call, given below, can be used to construct temporal basis functions; note that now `manifold = real_line()`, and we choose Gaussian basis functions instead (in which case `scale` represents the standard deviation). As in the spatial case, other basis functions (such as bisquare) can also be used.

```
R> G_temporal <- local_basis(manifold = real_line(), loc = matrix(seq(2,
+   28, by = 4)), scale = rep(3, 7), type = "Gaussian")
```

The generated basis functions can be visualized using `show_basis`; see Figure 20. The spatio-temporal basis functions are then constructed using the function `TensorP` as follows:

```
R> G <- TensorP(G_spatial, G_temporal)
```

The object `G` can be subsequently used for constructing SRE models, as in the spatial case. Note that since we have nine spatial basis functions and seven temporal basis functions, we have 63 spatio-temporal basis functions in total. Care should be taken that the total number of basis functions does not become prohibitively large (say > 4000).

6.2. Global prediction of column-averaged carbon dioxide from OCO-2

The NASA OCO-2 satellite was launched on 2014-07-02, and it produces between 100,000 and 300,000 usable retrievals per day. Between the beginning of October 2014 and the end of February 2017, the satellite produced around 100 million retrievals. The specific data product we used was the OCO-2 Data Release 7R Lite File Version B7305Br ([OCO-2 Science Team, Gunson, and Eldering 2015](#)). Following pre-processing, we reduced the number of data entries in the product to around 50 million. Each retrieval produces a number of variables; in this example, we consider the most commonly used one, XCO_2 , the column-averaged mole fraction in ppm. Obtaining reliable global predictions of XCO_2 does not require consideration of all the data simultaneously. The atmosphere mixes quickly within hemispheres, and temporal correlation-length scales are on the order of days. Hence, we consider the OCO-2 data in a moving-window of 16 days, and for each 16 days we fit a spatio-temporal SRE model in order to obtain a global prediction of XCO_2 in the middle (i.e., the 8th day) of the window. We use this moving-window approach to obtain daily XCO_2 global prediction and prediction errors, between 2014-10-01 and 2017-03-01.

We first put the OCO-2 data into an ‘STIDF’ object, before using the function `auto_BAUs` to construct spatio-temporal voxels. The following code constructs gridded BAUs and instructs **FRK** to use 1 day as the smallest temporal unit and to limit the latitude grid to the minimum and maximum latitude of the OCO-2 data locations, rounded to the nearest degree.

```
R> BAUs <- auto_BAUs(manifold = STsphere(), data = STobj,
+   type = "grid", tunit = "days", cellsize = c(1, 1, 1),
+   ylim = c(floor(min(oco2data$lat)), ceiling(max(oco2data$lat))),
+   xlim = c(-180, 180))
```

The code given above generated around 45,000 BAUs per day, for a total of around 720,000 BAUs per 16-day batch. We then constructed 396 spatial basis functions on the globe using

```
R> G_spatial <- auto_basis(manifold = sphere(), data = as(STobj,
+   "Spatial"), nres = 3, type = "bisquare", isea3h_lo = 1)
```

and eight temporal basis functions, one basis function for each two-day period, using

```
R> G_temporal <- local_basis(manifold = real_line(), loc = matrix(seq(1,
+   16, by = 2)), scale = rep(4, 8), type = "bisquare")
```

Finally we used **TensorP** to obtain a set of 3,168 spatio-temporal basis functions.

Following pre-processing, we had about one million usable soundings per 16-day window. However, several of these are in quick succession and thus also in close proximity to each other, so that they fall within the same spatio-temporal BAU. We therefore keep the flag `average_in_BAU` set to `TRUE` when calling `SRE` as in Section 4.2. Following averaging, the

number of observations per 16-day window reduces by a factor of about 100. We did not need to estimate the measurement-error variance σ_ϵ^2 in this case, since measurement-error variances are provided with the OCO-2 data. However, we forced all measurement-error standard deviations that were below 2 ppm to be equal to 2 ppm, since the reported values are likely to be optimistic. The total time needed to fit and predict with **FRK** in a single 16-day window was about 1 hour on a standard desktop computer.

In Figure 21 we show the measurement locations and values for the 16 days, and we show the central day of the 16-day window centered on 2016-09-08. Predictions and prediction standard errors for the central day are shown in Figure 22. Note how the error map reflects the data coverage over the entire 16-day window and not just the day at the center of the window. Prediction standard error maps on other days clearly show when the satellite is only taking readings over the ocean and when it is not taking any readings due to instrument reset or satellite manoeuvres. An animation showing these and other interesting features of predicted column-averaged CO₂ (i.e., XCO₂) between 2014-10-01 and 2017-03-01 is available at <https://www.youtube.com/watch?v=wEws67WXvkY>.

7. Future work

There are a number of useful features that could be implemented in future versions of **FRK**, some of which are listed below:

- Currently, **FRK** is designed to work with local basis functions having an analytic form. However, the package could also accommodate basis functions that have no known functional form, such as empirical orthogonal functions (EOFs) and classes of wavelets defined iteratively; future work will attempt to incorporate the use of such basis functions. Vanilla FRK (FRK-V), where the entire positive-definite matrix **K** is estimated, is particularly suited to the former (EOF) case where one has very few basis functions that explain a considerable amount of observed variability.
- There is currently no component of the model that caters for sub-BAU process variation, and each datum that is point-referenced is mapped onto a BAU. Going below the BAU scale is possible, and intra-BAU correlation can be incorporated if the covariance function of the process at the sub-BAU scale is known (Wikle and Berliner 2005).
- Most work and testing in **FRK** has been done on the real line, the 2D plane, and the surface of the sphere (\mathbb{S}^2). Other manifolds can be implemented since the SRE model always yields a valid spatial covariance function, no matter the manifold. Some, such as the 3D hyperplane, are not too difficult to construct. Ultimately, it would be ideal if the package would allow the user to specify his/her own manifold, along with a function that can compute the appropriate distances on the manifold.
- Currently, only the EM algorithm is implemented and hence the argument `method = "EM"` is implicit in the function `SRE.fit`. The EM algorithm has been seen to be slow to converge to a local maximum in other contexts (e.g., McLachlan and Krishnan 2007). Other methods for finding maximum, or restricted maximum, likelihood estimates for the SRE model (e.g., Tzeng and Huang 2018) will be considered for future versions of **FRK**.

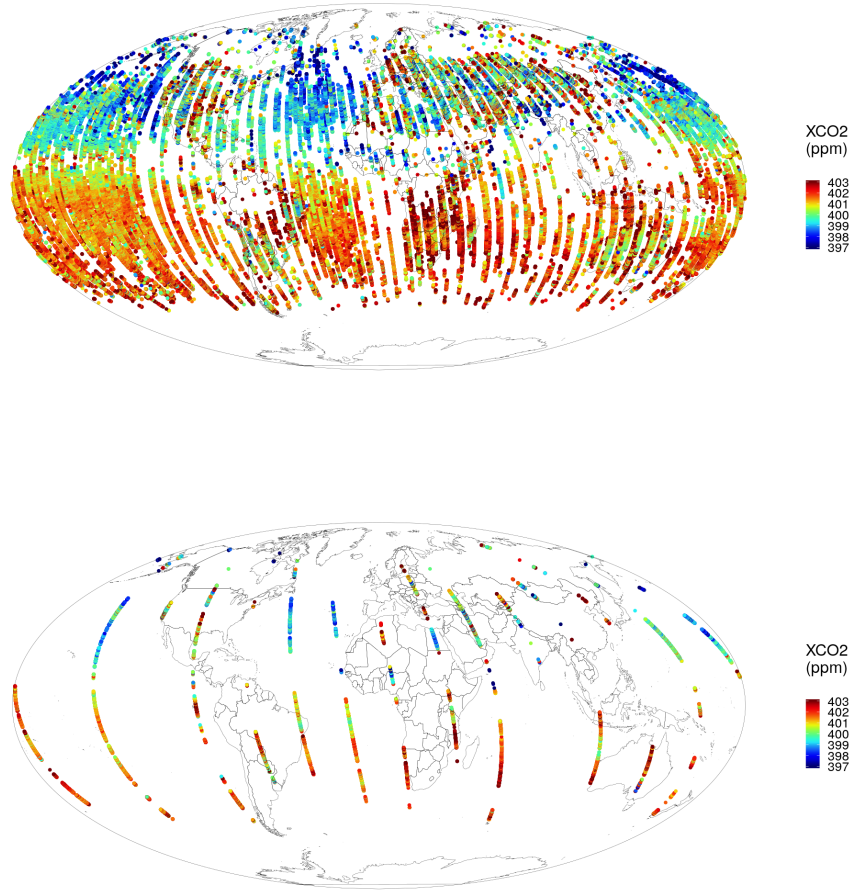
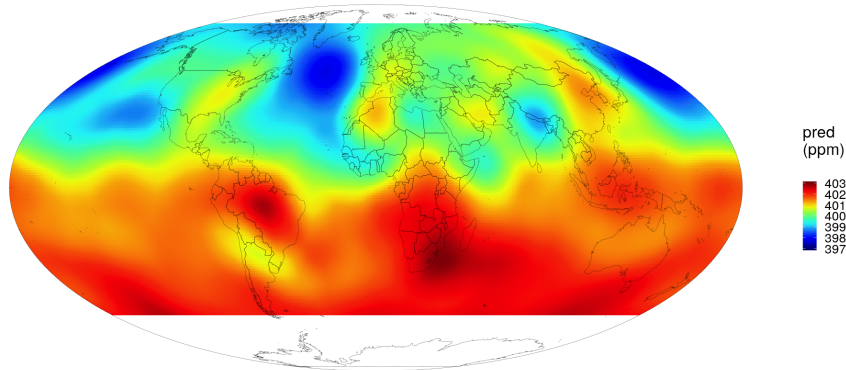


Figure 21: OCO-2 measurements of column-averaged CO₂ (XCO₂) in ppm. (Top panel) Data in the 16-day window from 2016-08-31 to 2016-09-15. (Bottom panel) Data from 2016-09-08.

- Although designed for very large data, **FRK** begins to slow down when several hundreds of thousands of data points are used. The flag `average_in_BAU` can be used to summarize the data and hence reduce the size of the dataset, however it is not always obvious how the data should be summarized (and whether one should summarize them in the first place). Future work will aim to provide the user with different options for summarizing the data.
- Currently, in **FRK**, all BAUs are assumed to be of equal area even if they are not. Unequal BAU area (see, for example, the lon-lat grid shown in Figure 13) is important when aggregating the process or the predictions. In **FRK** there is the option to use equal-area icosahedral grids on the surface of the sphere, and regular grids on the real line and the plane for when areal data or large prediction regions are used. (Note that in Section 6.2 an equal-area grid was not used, but also note that we did not spatially aggregate our results and that our data were point-referenced).

2016-09-08 Prediction



2016-09-08 Uncertainty

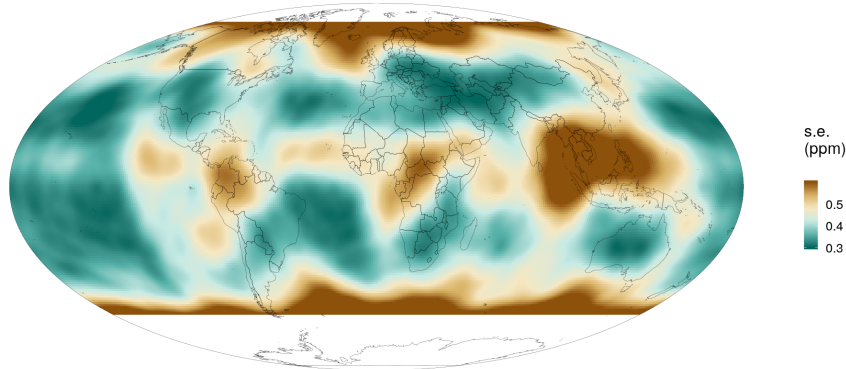


Figure 22: Prediction and prediction standard errors of column-averaged CO_2 (XCO_2) on 2016-09-08 obtained using FRK. (Top panel) Prediction of \mathbf{Y}_P in ppm. (Bottom panel) Prediction standard errors in ppm.

In conclusion, the package **FRK** is designed to provide core functionality for spatial and spatio-temporal prediction with large datasets. The low-rank model used by the package has validity (accurate coverage) in a big-data scenario when compared to high-rank models that do not explicitly cater for fine-scale variation. However, it is likely to be less statistically efficient (larger root mean squared prediction errors) than other methods when data density is high and the basis functions are unable to capture the spatial variability.

FRK is available on the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/package=FRK>. Its development page is <https://github.com/andrewzm/FRK>. Users are encouraged to report any bugs or issues relating to the package on this web page.

Acknowledgments

Package development was facilitated with **devtools** (Wickham, Hester, and Chang 2021); this paper was compiled using **knitr** (Xie 2015); and package testing was carried out using **testthat** (Wickham 2011) and **covr** (Hester 2020). We would like to thank the reviewers for their feedback on the original submission and Esri staff with whom the classical factorial design of Section 4.1 was discussed. We thank Jonathan Rougier for helpful comments on the manuscript; Doug Nychka and Finn Lindgren for advice on setting up **LatticeKrig** and **INLA**, respectively, to model data from a process with exponential covariance function; Clint Shumack for using **FRK** to analyze the OCO-2 data; and Enki Yoo and Behzad Kianian for providing valuable feedback on the package. AZM’s research was partially supported by a Senior Research Fellowship from the University of Wollongong (2015–2017) and an Australian Research Council Discovery Early Career Research Award, DE180100203 (2018). NC’s research was supported by an Australian Research Council Discovery Project, DP150104576, and by the US National Science Foundation under NSF grant SES-1132031 funded through the NSF-Census Research Network (NCRN) program.

References

- Banerjee S, Gelfand AE, Finley AO, Sang H (2008). “Gaussian Predictive Process Models for Large Spatial Data Sets.” *Journal of the Royal Statistical Society B*, **70**(4), 825–848. doi:10.1111/j.1467-9868.2008.00663.x.
- Bates D, Maechler M (2021). **Matrix: Sparse and Dense Matrix Classes and Methods**. R package version 1.3-3, URL <http://CRAN.R-project.org/package=Matrix>.
- Bivand R (2021). *CRAN Task View: Analysis of Spatial Data*. Version 2021-05-20, URL <https://CRAN.R-project.org/view=Spatial>.
- Bivand RS, Pebesma E, Gómez-Rubio V (2013). *Applied Spatial Data Analysis with R*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-1-4614-7618-4.
- Chahine MT, Pagano TS, Aumann HH, Atlas R, Barnett C, Blaisdell J, Chen L, Divakarla M, Fetzer EJ, Goldberg M, Gautier C, Granger S, Hannon S, Irion FW, Kakar R, Kalnay E, Lambrigtsen BH, Lee SY, Marshall JL, McMillan WW, McMillin L, Olsen ET, Revercomb H, Rosenkranz P, Smith WL, Staelin D, Larrabee Strow L, Susskind J, Tobin D, Wolf W, Zhou L (2006). “AIRS: Improving Weather Forecasting and Providing New Data on Greenhouse Gases.” *Bulletin of the American Meteorological Society*, **87**(7), 911–926. doi:10.1175/bams-87-7-911.
- Cressie N, Johannesson G (2006). “Spatial Prediction for Massive Data Sets.” In *Mastering the Data Explosion in the Earth and Environmental Sciences: Proceedings of the Australian Academy of Science Elizabeth and Frederick White Conference*, pp. 1–11. Australian Academy of Science, Canberra, Australia.
- Cressie N, Johannesson G (2008). “Fixed Rank Kriging for Very Large Spatial Data Sets.” *Journal of the Royal Statistical Society B*, **70**(1), 209–226. doi:10.1111/j.1467-9868.2007.00633.x.

- Cressie N, Shi T, Kang EL (2010). “Fixed Rank Filtering for Spatio-Temporal Data.” *Journal of Computational and Graphical Statistics*, **19**(3), 724–745. doi:10.1198/jcgs.2010.09051.
- Davis T (2021). **sparseinv**: Sparse Inverse Subset. URL <https://github.com/DrTimothyAldenDavis/SuiteSparse>.
- Finley AO, Banerjee S, Carlin BP (2007). “spBayes: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models.” *Journal of Statistical Software*, **19**(4), 1–24. doi:10.18637/jss.v019.i04.
- Finley AO, Banerjee S, Gelfand AE (2015). “spBayes for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models.” *Journal of Statistical Software*, **63**(13), 1–28. doi:10.18637/jss.v063.i13.
- Fuglstad GA, Simpson D, Lindgren F, Rue H (2019). “Constructing Priors That Penalize the Complexity of Gaussian Random Fields.” *Journal of the American Statistical Association*, **114**(525), 445–452. doi:10.1080/01621459.2017.1415907.
- Gneiting T, Balabdaoui F, Raftery AE (2007). “Probabilistic Forecasts, Calibration and Sharpness.” *Journal of the Royal Statistical Society B*, **69**(2), 243–268. doi:10.1111/j.1467-9868.2007.00587.x.
- Gneiting T, Raftery AE (2007). “Strictly Proper Scoring Rules, Prediction, and Estimation.” *Journal of the American Statistical Association*, **102**(477), 359–378. doi:10.1198/016214506000001437.
- Henderson HV, Searle SR (1981). “On Deriving the Inverse of a Sum of Matrices.” *SIAM Review*, **23**(1), 53–60. doi:10.1137/1023004.
- Hester J (2020). **covr**: Test Coverage for Packages. R package version 3.5.1, URL <https://CRAN.R-project.org/package=covr>.
- Kang EL, Cressie N (2011). “Bayesian Inference for the Spatial Random Effects Model.” *Journal of the American Statistical Association*, **106**(495), 972–983. doi:10.1198/jasa.2011.tm09680.
- Kang EL, Liu D, Cressie N (2009). “Statistical Analysis of Small-Area Data Based on Independence, Spatial, Non-Hierarchical, and Hierarchical Models.” *Computational Statistics & Data Analysis*, **53**(8), 3016–3032. doi:10.1016/j.csda.2008.07.033.
- Katzfuss M, Cressie N (2011). “Spatio-Temporal Smoothing and EM Estimation for Massive Remote-Sensing Data Sets.” *Journal of Time Series Analysis*, **32**(4), 430–446. doi:10.1111/j.1467-9892.2011.00732.x.
- Katzfuss M, Hammerling D (2017). “Parallel Inference for Massive Distributed Spatial Data Using Low-Rank Models.” *Statistics and Computing*, **27**(2), 363–375. doi:10.1007/s11222-016-9627-4.
- Kleiber W (2016). “High Resolution Simulation of Nonstationary Gaussian Random Fields.” *Computational Statistics & Data Analysis*, **101**, 277–288. doi:10.1016/j.csda.2016.03.005.

- Lindgren F, Rue H (2015). “Bayesian Spatial Modelling with R-INLA.” *Journal of Statistical Software*, **63**(19), 1–25. doi:[10.18637/jss.v063.i19](https://doi.org/10.18637/jss.v063.i19).
- McLachlan G, Krishnan T (2007). *The EM Algorithm and Extensions*. John Wiley & Sons, Hoboken.
- Nguyen H, Cressie N, Braverman A (2012). “Spatial Statistical Data Fusion for Remote Sensing Applications.” *Journal of the American Statistical Association*, **107**(499), 1004–1018. doi:[10.1080/01621459.2012.694717](https://doi.org/10.1080/01621459.2012.694717).
- Nguyen H, Katzfuss M, Cressie N, Braverman A (2014). “Spatio-Temporal Data Fusion for Very Large Remote Sensing Datasets.” *Technometrics*, **56**(2), 174–185. doi:[10.1080/00401706.2013.831774](https://doi.org/10.1080/00401706.2013.831774).
- Nychka D, Bandyopadhyay S, Hammerling D, Lindgren F, Sain S (2015). “A Multiresolution Gaussian Process Model for the Analysis of Large Spatial Datasets.” *Journal of Computational and Graphical Statistics*, **24**(2), 579–599. doi:[10.1080/10618600.2014.914946](https://doi.org/10.1080/10618600.2014.914946).
- Nychka D, Hammerling D, Sain S, Lenssen N, Smirniotis C, Iverson M (2019). **LatticeKrig**: Multiresolution Kriging Based on Markov Random Fields. R package version 8.4, URL <https://CRAN.R-project.org/package=LatticeKrig>.
- OCO-2 Science Team, Gunson M, Eldering A (2015). *OCO-2 Level 2 Bias-Corrected XCO₂ and Other Select Fields from the Full-Physics Retrieval Aggregated as Daily Files, Retrospective Processing V7r*. Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), URL https://hpc.niasra.uow.edu.au/ckan/dataset/oco2_l2_lite_fp_v7r.
- Pebesma E (2012). “**spacetime**: Spatio-Temporal Data in R.” *Journal of Statistical Software*, **51**(7), 1–30. doi:[10.18637/jss.v051.i07](https://doi.org/10.18637/jss.v051.i07).
- Pebesma EJ (2004). “Multivariable Geostatistics in S: The **gstat** Package.” *Computers & Geosciences*, **30**(7), 683–691. doi:[10.1016/j.cageo.2004.03.012](https://doi.org/10.1016/j.cageo.2004.03.012).
- Rasmussen CE, Williams CKI (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge. doi:[10.7551/mitpress/3206.001.0001](https://doi.org/10.7551/mitpress/3206.001.0001).
- R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Sampson PD, Guttorp P (1992). “Nonparametric Estimation of Nonstationary Spatial Covariance Structure.” *Journal of the American Statistical Association*, **87**(417), 108–119. doi:[10.1080/01621459.1992.10475181](https://doi.org/10.1080/01621459.1992.10475181).
- Schlather M, Malinowski A, Menck PJ, Oesting M, Strokorb K (2015). “Analysis, Simulation and Prediction of Multivariate Random Fields with Package **RandomFields**.” *Journal of Statistical Software*, **63**(8), 1–25. doi:[10.18637/jss.v063.i08](https://doi.org/10.18637/jss.v063.i08).
- Shi T, Cressie N (2007). “Global Statistical Analysis of MISR Aerosol Data: A Massive Data Product from NASA’s Terra Satellite.” *Environmetrics*, **18**(7), 665–680. doi:[10.1002/env.864](https://doi.org/10.1002/env.864).

- Stein ML (2008). “A Modeling Approach for Large Spatial Datasets.” *Journal of the Korean Statistical Society*, **37**(1), 3–10. doi:[10.1016/j.jkss.2007.09.001](https://doi.org/10.1016/j.jkss.2007.09.001).
- Tzeng S, Huang HC (2018). “Resolution Adaptive Fixed Rank Kriging.” *Technometrics*, **60**(2), 198–208. doi:[10.1080/00401706.2017.1345701](https://doi.org/10.1080/00401706.2017.1345701).
- Wang Q, Zhang X, Zhang Y, Yi Q (2013). “AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on X86 CPUs.” In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. Denver.
- Wickham H (2011). “**testthat**: Get Started with Testing.” *The R Journal*, **3**(1), 5–10. doi:[10.32614/rj-2011-002](https://doi.org/10.32614/rj-2011-002).
- Wickham H, Hester J, Chang W (2021). **devtools**: *Tools to Make Developing R Packages Easier*. R package version 2.4.1, URL <https://CRAN.R-project.org/package=devtools>.
- Wikle CK, Berliner LM (2005). “Combining Information across Spatial Scales.” *Technometrics*, **47**(1), 80–91. doi:[10.1198/004017004000000572](https://doi.org/10.1198/004017004000000572).
- Xie Y (2015). *Dynamic Documents with R and knitr*. 2nd edition. Chapman & Hall/CRC, Boca Raton.
- Zammit-Mangion A (2018). **sparseinv**: *Computation of the Sparse Inverse Subset*. R package version 0.1.3, URL <https://CRAN.R-project.org/package=sparseinv>.
- Zammit-Mangion A (2020). **dggrids**: *Discrete Global Grids in Data Frame Format for Use in R (and Fixed Rank Kriging in Particular)*. R package version 0.1.0, URL <https://github.com/andrewzm/dggrids>.
- Zammit-Mangion A, Cressie N, Shumack C (2018). “On Statistical Approaches to Generate Level 3 Products from Satellite Remote Sensing Retrievals.” *Remote Sensing*, **10**(1), 155. doi:[10.3390/rs10010155](https://doi.org/10.3390/rs10010155).
- Zammit-Mangion A, Rougier J (2018). “A Sparse Linear Algebra Algorithm for Fast Computation of Prediction Variances with Gaussian Markov Random Fields.” *Computational Statistics & Data Analysis*, **123**, 116–130. doi:[10.1016/j.csda.2018.02.001](https://doi.org/10.1016/j.csda.2018.02.001).
- Zammit-Mangion A, Rougier J, Schön N, Lindgren F, Bamber J (2015). “Multivariate Spatio-Temporal Modelling for Assessing Antarctica’s Present-Day Contribution to Sea-Level Rise.” *Environmetrics*, **26**(3), 159–177. doi:[10.1002/env.2323](https://doi.org/10.1002/env.2323).
- Zammit-Mangion A, Sainsbury-Dale M (2021). **FRK**: *Fixed Rank Kriging*. R package version 2.0.0, URL <https://CRAN.R-project.org/package=FRK>.
- Zammit-Mangion A, Sanguinetti G, Kadirkamanathan V (2012). “Variational Estimation in Spatiotemporal Systems from Continuous and Point-Process Observations.” *IEEE Transactions on Signal Processing*, **60**(7), 3449–3459. doi:[10.1109/tsp.2012.2191966](https://doi.org/10.1109/tsp.2012.2191966).
- Zhuang L, Cressie N (2014). “Bayesian Hierarchical Statistical SIRS Models.” *Statistical Methods & Applications*, **23**(4), 601–646. doi:[10.1007/s10260-014-0280-9](https://doi.org/10.1007/s10260-014-0280-9).

Affiliation:

Andrew Zammit-Mangion

National Institute for Applied Statistics Research Australia (NIASRA)

School of Mathematics and Applied Statistics

University of Wollongong

Wollongong, Australia

E-mail: azm@uow.edu.au

URL: <https://andrewzm.wordpress.com/>